Design and Implementation of a Popular Science Game for VR Medical Surgical Instruments Based on Unity3D

Liyan Liu
Shandong University of Political Science and Law
Jinan, Shandong Province, China
Liuliyansd@163.com

***Abstract*—This article focuses on the science popularization of medical surgical instruments, analyzes the current situation of medical science popularization games, concludes the necessity of the existence of this game, realizes the interesting science popularization of medical instruments, and achieves the goal of using games to promote science popularization and vividly popularize knowledge in a virtual environment. The innovation lies in the fact that this game is not a game in the traditional sense, but rather a game of popular science and education, which promotes the cultivation of autonomous and spontaneous learning; Most games are played as doctors, while this game is played as an instrument nurse, allowing players to understand this profession and respect every medical worker more; It provides a new perspective for medical education and can construct a new popular science teaching model to better cultivate experimental and observation abilities.**

Keywords—VR; Interesting science popularization; Unity game engine; A new perspective on medical education

I. SOCIAL BACKGROUND AND DEMAND ANALYSIS OF GAME DEVELOPMENT

In the context of the current curriculum reform of basic education in China and the integration of information technology and teaching, it is necessary to constantly update learning methods in order to adapt to new teaching concepts. Information technology has gradually expanded from the film and television entertainment industry to the field of education. Visual learning resources and interactive learning scenarios based on information technology can help learners conduct situational learning, experiential learning, and game learning, thereby improving learning efficiency[1]. In response to this problem, this design is based on Unity 3D, combined with current popular game development engine technology, to design a popular science game for medical surgery main instruments based on Unity 3D. This game is designed to develop a serious medical game, targeting new medical personnel, medical university students, and medical fans. It is a serious game based on Unity 3D[2].

In order to make the game more responsive to the audience's preferences, developers collected relevant information through questionnaires. On WeChat and QQ, a total of 247 questionnaires were distributed, 216 questionnaires were recovered, and 203 valid questionnaires were valid, with an effective rate of 82.2%. After collecting and sorting out the results, on the one hand, we have explored the public's understanding and acceptance of the medical science popularization game, and on the other hand, we have explored everyone's expectations and thoughts on various aspects of the game[3]. Based on the summary of the survey results and relevant data, as well as the complexity and strong professionalism of medical knowledge, this game aims to create a relaxed and interesting learning atmosphere for students. Therefore, according to the needs of the user population, this game must meet the following requirements:

- Ensure that the design of the content in the system is scientific, professional, and targeted.
- The interface is beautiful, interactive, and clear, easy to understand and use, improving the user's experience of using.
- Strive to pursue the realism of scenes and devices to enhance the user's sense of substitution. And the system must be able to adjust to user behavior.
- Providing popular science education to users based on rich and efficient teaching content can make it easier for users to acquire professional skills.

II. THE DEVELOPMENT VALUE OF VR MEDICAL SURGERY MAIN EQUIPMENT SCIENCE POPULARIZATION GAME BASED ON UNITY3D

- From the perspective of game design, develop a serious game with medical significance. Its application is not only limited to medical workers, but also targeted at the general population. It should not only have the role of vocational training, but also have the role of popularizing medical knowledge[4].
- This serious game attaches great importance to the basic design concepts of the game in its design, drawing on many medical videos and data, and paying attention to aesthetics, as well as in-depth learning of relevant professional skills, in order to make this game more professional and interesting.
- While learning professional knowledge, it is necessary to conduct some operational skills training, organically linking it with reality, and more suitable for today's teaching and training trends. This is a reinforcement and

979-8-3503-1752-7/23/$31.00 ©2023 IEEE

supplement to traditional education, and it is truly "teaching with pleasure"[5].

III. Development Strategy of VR Medical Surgery Main Equipment Science Popularization Game Based on Unity3D

A. Design ideas

This game is very similar to virtual simulation experiments, which can improve the traditional experimental teaching mode, provide effective modeling tools for the teaching of basic medical experiments, a large number of model resources, and simulate the construction of laboratories. Using this technology can achieve many experiments in basic medicine, which can enrich the experimental teaching content of basic medicine. At the same time, on this basis, the quality of experimental teaching in medical courses has been further improved, allowing teachers to enrich their teaching tools and students to have a stronger interest in learning[6].

There are many types of virtual experiments in basic medicine, involving many modules and a wide range of content. Through this game, the educational effects for students will be richer. Through virtual simulation of experimental teaching, students' enthusiasm for learning can be greatly improved, thereby expanding the scope of the experiment infinitely and breaking the limitations of time and space in the classroom. The establishment of the virtual simulation experiment model has opened up a new space for medical college students and brought new opportunities for medical teachers' teaching work. At the same time, it also provides strong support for innovative medical experimental teaching models[7].

The goal of this game is to develop a medical science popularization game based on the Unity engine. The background of the game is an ongoing surgery, while the player, as an instrument nurse, correctly selects the instruments required by the doctor during the surgery process. Because it is related to medicine, this is a relatively serious game. It is necessary to simulate the real environment as much as possible and restore the scene requirements as highly as possible. The scene model and device model of the entire game should follow the principle of consistency, treat the modeling with a rigorous attitude, and meet the image expectations of relevant medical enthusiasts[8].

B. functional design

Based on the analysis of the results of the questionnaire, the overall framework of the entire game is constructed, mainly including the following four modules.

1) UI interface processing

The composition of a game not only lies in the organic combination of various scenarios, but also requires the participation of the UI interface to connect various scenarios. The UI interface of this game mainly includes:

- suggestion submission interface display: users can provide ideas and suggestions for problems in the game for developers to make better improvements;
- Selection interface display: used to enter other interfaces or scenarios;
- List of surgical instruments: List the required instruments, convenient and intuitive;
- The expansion option displays: for players to learn more basic knowledge and additional knowledge.

The UI interface is mainly used to provide auxiliary functions for users, making the game experience more humanized and humane[9].

2) Scene control processing

This game provides users with a simulated medical environment, where the game is played on a scene by scene basis. In the surgical environment, the models in each type of surgery include surgical instrument models and environmental background models that are necessary for surgical execution. Result evaluation is an additional function. Provide the most important feedback for the player's game process by eliciting results based on the player's performance during the game process.

3) Video processing

In order to enable players to gain more knowledge, the game will provide options for players to watch surgical related expansion videos. Although video processing is an auxiliary function of game design, video is a good interactive method that plays a powerful role in enhancing the game experience.

4) Game Description

The game description section provides medical terminology explanations for players to view. The game development has functional scalability, providing a good environment for further optimization and improvement of the system.

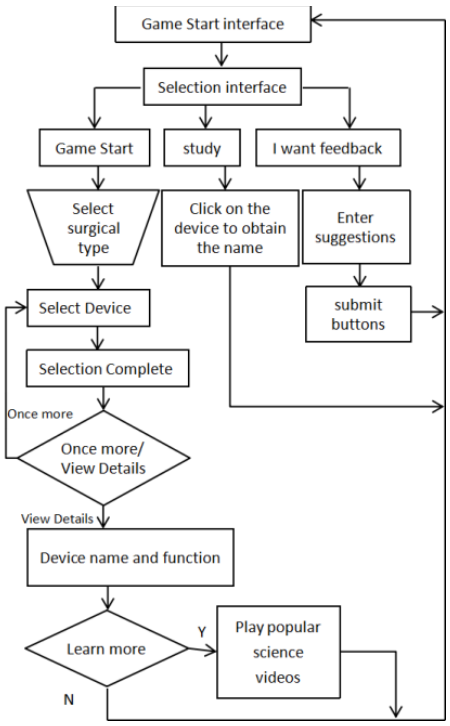


Fig. 1. Game operation flowchart

C. Overall framework

This game is a VR medical science popularization game based on the Unity engine. The background of the game is an ongoing surgery, and the player, as an instrument nurse, correctly selects the instruments required by doctors during the surgery process. Because it is related to medicine, this is a relatively serious game. It is necessary to simulate the real environment as much as possible and restore the scene requirements as highly as possible. The scene model and device model of the entire game should follow the principle of consistency, treat the modeling with a rigorous attitude, and meet the image expectations of relevant medical enthusiasts. An overview of the operation process is shown in Figure 1.

IV. GAME DESIGN

A. model design

Unity 3D technology itself cannot provide 3D modeling, so specialized modeling software must be available to complete it. This scheme uses the currently popular 3D Max animation rendering and rendering software. During production, try to restore the realism of the device, as shown in Figure 2.

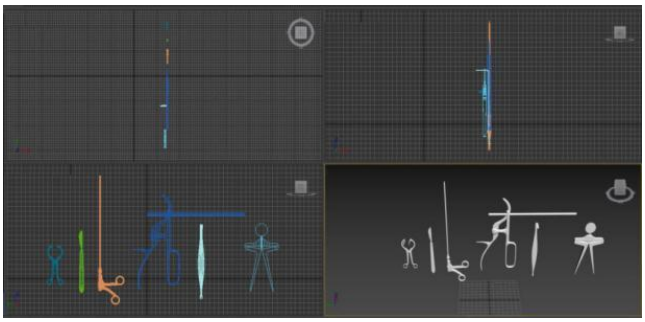


Fig. 2. From left to right, there are large blood vessel forceps, knife handle, grip forceps, tonsil entrapment device, hook opener

B. Scene Design

1) Scene resource design

The continuity of the game requires the splicing and combination of scenes, which is particularly important for the design of scenes. After entering the main interface, the design scenario is shown in Figure 3, and the hierarchical scenario resource interface is shown in Figure 4.

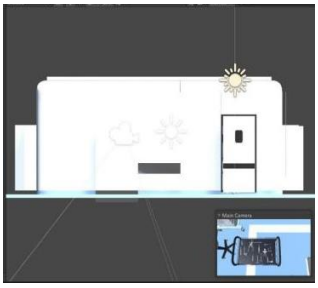


Fig. 3. Main perspective Unity 3D scene

2) Background resource design

The background related resource design in the game is the model and special effects that can be seen in the scene, as shown in Figure 5.

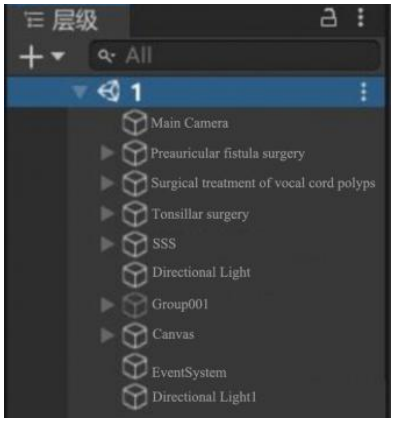


Fig. 4. Parent-child relationships included in the scene level view

Fig. 5. Operating Room Scene

C. Establishment of development environment

1) Game Scene Import

Use 3D Max to model the equipment and scene required for the game, save it in .fpx format, as shown in Figure 6, import it to Unity, and construct the scene in it.

2) Implementation of camera placement, rotation, and zoom perspective functions

The camera is indispensable in the game world built by

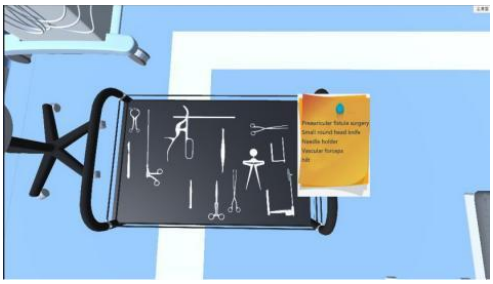


Fig. 6. Scenario setup

Unity. Without the camera, the game world would be completely dark, just like our eyes, used to observe the created game world.The placement method of the camera is as follows (the effect is shown in Figure 7):

- Right click on the hierarchy panel to create a camera.
- If the camera is selected, a preview window will appear in the Scene view.
- Change and change the position and angle of the camera according to the axial direction.
- You need to align the camera to the current view, so that Scene and Game can maintain consistent viewing angles and locations in the game.

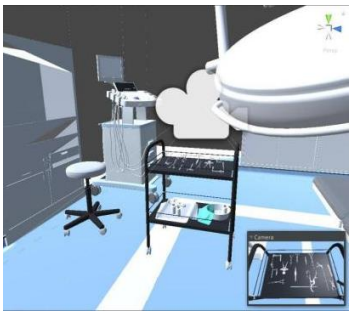


Fig. 7. Camera placement

This game requires a free angle of view to rotate, shrink, and zoom in to carefully observe the shape of the device. These operations need to be achieved by manipulating the camera.

V. CONCLUSIONS

Based on the previous investigation, this article proposed a design scheme for medical games, modeled the operating room scene and surgical instruments therein, and applied it to Unity 3D using C # language, achieving a relatively complete effect. Through the design, development and promotion of this game, we hope to make certain contributions to the development of medical education. This design is a combination of serious games and interdisciplinary research in medicine. However, there are also some deviations between the actual development and expectations of this game. The following three points are summarized for future research work:

- There is still room for improvement in the design and functional implementation of game scenarios. Only some relatively simple interactions have been achieved, but a more humanized teaching experience has not yet been achieved. The focus of future work is to strengthen the construction of game scenes, increase the functionality of the game, and enhance the development technology of Unity 3D.
- Game operations are only developed on PC platforms, and some attempts will be made on mobile phones in the future to reduce the limitations of the usage area and improve the "anytime, anywhere" nature of learning.
- Strengthen large-scale questionnaire surveys and feedback, and increase diversity analysis.

In the future of rapid technological progress, serious medical games are bound to become a highlight in the medical field. Advanced technology makes game design more diverse, and it also makes serious medical games more credible.

ACKNOWLEDGMENT

This paper is supported by the undergraduate teaching reform research project of Shandong Province "Research and Practice on the Integration of Ideological and Political Education and Innovation and Entrepreneurship Education" (M2021262) and the scientific research project of Shandong University of Political Science and Law (Innovation and Entrepreneurship Project) "Research on the Implementation Path of Ideological and Political Integration Course of Innovation and Entrepreneurship Education in Colleges and Universities" (2022CQ02Z).

REFERENCES

[1] Wang Pingan, “Medical Image Analysis and Surgical Simulation: Applications of Artificial Intelligence and Virtual Reality in Medicine,” Optics and Optoelectronics Technology, 2021,19 (06): 1-10. DOI: 10.19519/j.cnki. 1672-3392.2021.06.001.

[2] Shi Xiaowei, Yuan Hui, Lu Mingxuan, Cai Jiahui, and Zhang Xianzeng, “Research Status and Progress of Virtual Reality Technology in the Medical Field,” Progress in Laser and Optoelectronics, 2020,57 (01): 66-75.

[3] Janarthanan Balakrishnan, Mark D. Griffiths, “Perceived Addictiveness of Smartphone Games: A Content Analysis of Game Reviews by Players”, International Journal of Mental Health and Addiction, Vol. 17, Issue 4, 2019, pp. 922-934.

[4] Wang Yongjie, Cui Libin, Yuan Xin, and Chen Xueming. “Application of virtual reality technology in clinical medicine teaching,” Medical Education Management, 2021,7 (01): 73-77.

[5] Rong Zhu and Yong Wang, "Design and Realization of Virtual Classroom," Journal of Advances in Information Technology, Vol. 3, No. 1, pp. 24-28, February, 2012.doi:10.4304/jait.3.1.24-28.

[6] Jining Han, Qiyu Zheng, and Yunan Ding, "Lost in Virtual Reality? Cognitive Load in High Immersive VR Environments," Journal of Advances in Information Technology, Vol. 12, No. 4, pp. 302-310, November 2021. doi: 10.12720/jait.12.4.302-310.

[7] Liu Tao, Research on key technologies of virtual surgery training system based on neurosurgery, Yunnan Normal University, 2021. DOI: 10.27459/d.cnki.gynfc.2021.000569.

[8] Wang Yingshu, Research on VR serious game design based on Unity3D technology, Changchun University of Technology, 2020. DOI: 10.27805/d.cnki.gccgy.2020.000699.

[9] Xu Xiuping, “Research on the development of children's puzzle games based on Android mobile platform”, Shanxi Youth, 2021 (02): 166-167.

论文收录引用

检 索 报 告

（检索日期：2024-01-19，检索责任人：李亚设）

委 托 人： 刘丽彦

委托人单位： 山东政法学院

发表论文单位： Shandong University of Political Science and Law, Shandong Province, Jinan, China

检 索 要 求： 刘丽彦发表的论文被 EI 收录

检 索 时 段： 2023 年至今

检 索 结 果

数据库	论文收录
EI Compendex	1 篇

声明：本次检索根据刘丽彦提供的论文目录，并按其提出的要求进行检索，详情见附件，若有不实，由委托人承担全部责任。

山东师范大学图书馆

（盖章）

二〇二四年一月二十一日

250014 山东师范大学图书馆
Tel: 0531-86180761 或 86181968
Email:375360605@qq.com

山东师范大学图书馆检索证明报告——附件

《工程索引》(EI)收录 1 篇

<RECORD 1>

Accession number:20240215357133
Title:Design and Implementation of a Popular Science Game for VR Medical Surgical Instruments Based on Unity3D
Authors:Liu, Liyan (1)
Author affiliation:(1) Shandong University of Political Science and Law, Shandong Province, Jinan, China
Corresponding author:Liu, Liyan(Liuliyansd@163.com)
Source title:16th International Conference on Advanced Computer Theory and Engineering, ICACTE 2023
Abbreviated source title:Int. Conf. Adv. Comput. Theory Eng., ICACTE
Part number:1 of 1
Issue title:16th International Conference on Advanced Computer Theory and Engineering, ICACTE 2023
Issue date:2023
Publication year:2023
Pages:118-121
Language:English
ISBN-13:9798350317534
Document type:Conference article (CA)
Conference name:16th International Conference on Advanced Computer Theory and Engineering, ICACTE 2023
Conference date:September 15, 2023 - September 17, 2023
Conference location:Hefei, China
Conference code:195295
Sponsor:Anhui Jianzhu University; IEEE Hefei Subsection
Publisher:Institute of Electrical and Electronics Engineers Inc.
DOI:10.1109/ICACTE59887.2023.10335331
Database:Compendex

委托人贡献:首作者、通讯作者

2023 4th Information Communication Technologies Conference

Design and Implementation of Mobile Balance Games Based on Mobile Platforms

Liyan Liu
Shandong University of Political Science and Law
JiNan, Shandong Province, China
Liuliyansd@163.com

***Abstract*—In the early stage of this paper, by studying the development status of mobile platforms, the evolution process of original games and the rapid development of mobile phones, in order to meet the needs of many users and adapt to the current scale of mobile phone users, we developed a set of balanced small games with strong entertainment and interactivity suitable for people of all levels - Dance of rattan.**

Keywords-Mobile platform, mobile game, balance game, unity

I. CURRENT SITUATION AND ANALYSIS OF MOBILE GAME BASED ON MOBILE PLATFORM

In recent years, the socio-economic level and scientific research technology have been developing at a very high speed. At the same time, mobile terminal technology has also been rapidly improved, and the mobile communication industry will move towards a real mobile information era. The use of shopping platforms, work platforms and common information platforms has become more and more extensive and frequent, which is essential for people's life and production activities. Accordingly, people's demand for mobile platform applications has gradually increased, and more and more attention has been paid to visual effects and interactive experience in use. Animation, as a means of expression, is more vivid and effective than pictures and words, and integrates sight and hearing. The convenience of mobile terminals and the effective combination of animation make the function of animation not only limited to viewing, but also become a vivid and wonderful way of expression. In mobile game design, the application of animation has brought more creative space for the game, which can express the original intention of the designer in a richer and more detailed way.

With the rapid development of smart phones, mobile game software business is also booming, and various mobile game businesses and development industries are growing. More mobile phone users hope to have leisure and entertainment through convenient and portable devices after work and study. Although there are a variety of game versions on the market, the market for balance games is still quite large. Its special feature is that it can attract people to go deeper and love it. With the increasing difficulty of the game, its stimulation is also stronger. The advantage of this game is that it is simple and easy. For people who are busy outside, it is impossible to spend a lot of time on entertainment. Large-scale games are not feasible. Such small games just meet this demand.

II. DESIGN OF MOBILE GAME SYSTEM OF "DANCE OF RATTAN"

A. Design Inspiration

Create a new trend in the game, "Dance of rattan" is a dreamlike flying character balance game. The difficulty of the game is not too high. It is mainly to obtain gold coins to maintain balance, avoid obstacles and rolling stone attacks, achieve customs clearance at the destination, and let you experience different game experiences at the same time. The vine in the game is a symbol of people's continuous pursuit. Just like modern people can only get what they want if they keep working hard. The rolling stones in the game are also hidden collisions. Just like the various obstacles in life now, if you encounter problems, you can only solve them through your unremitting efforts.

B. System Architecture

UI module: open screen background image, option Button skin, prompt interface, pop-up interface, interface layout, and local model.

Plot module: interactive game content, plot logic.

Main functions: system initialization settings, interface and map settings, game data database settings, game process control, game archiving, and game execution.

Model module: It mainly includes two modules: the game main interface module and the game control module. The game main interface module mainly includes the game graphic area interface, the game start button, the pause button, the main menu button, the volume control button, the forward, backward, jump button and the advertising button; The game control module mainly completes the functions of game start, pause, return to the main menu, volume adjustment, forward, backward, jump and advertising.

Map generation: We have introduced the mature Unity plug-in Ferr to generate maps.

Code module: realize the functions in the game by calling the modules in Unity3D, UI, Animator, Audio Source, etc.

C. System Development Environment

Operating system: computer, Win10 operating system;

Software environment: Unity3D, Visual Studio 2019, Photoshop cc 2019;

978-1-6654-6257-0/23/$31.00 ©2023 IEEE

Development language: C #.

D. *Operation Process*

At the beginning of the game, the characters walk forward quickly in the jungle. When walking to the vine, pay attention to the balance bar on the top of the head to keep balance. If the balance bar is too far to the left or too far to the right, the game characters will lose their balance and fall off the vine. Therefore, the mobile phone gravity control should be used to help the characters keep balance in the game. During walking, there will be hidden trigger obstacles, that is, rolling stones on the ground, and enemies flying in the sky. At the same time, in order to increase the playability of the game, the function of eating gold coins is set for the game object. The more gold coins, the higher the score. What needs to be special is that the character needs to master the balance value on the vine. When the moving bead representing the balance moves to the left or right, the protagonist will fall off the cliff and die. The game process is shown in Fig. 1.

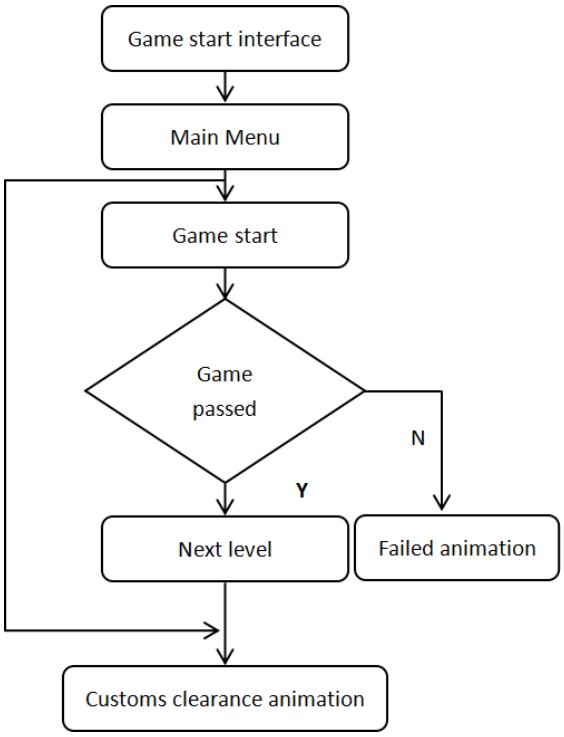


Fig. 1. Game flow chart

III. GAME IMPLEMENTATION

A. *Design of Game Background Database Module*

The game needs a large amount of file storage and a large capacity of database storage, so there are very high requirements for the game reading speed and file data storage. The data flow diagram of the system is shown in Fig. 2.

The main data structures involved in the game system are as follows:

- Archive function: used to record the player's current game progress, and its data structure is a sequential structure.
- Prop: used to improve the score of each role. Its data structure is an enumeration type.
- Enemies: The enemies (rolling stones, birds, etc.) that the character encounters in the game can get scores after hiding. Its data structure is a sequential structure.
- System information: the data structure is a sequential structure for the relevant music and background settings of the start interface.

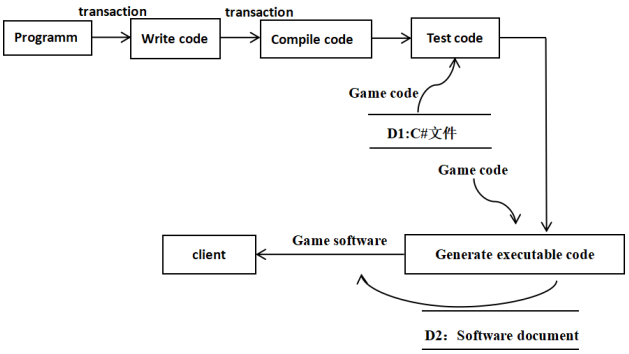


Fig. 2. Data flow diagrams

B. *Interface Design*

This design is based on a 2D horizontal balanced breakthrough game software of Unity. The game interface sequence: start the game - enter the game - save the game - end the game. In the interface rendering, the terrain is generated using the unity plug-in Ferr, and all other materials are hand-drawn, and generated and made into frame-by-frame animation using ps, create and other software.

Game menu interface: This interface is the initial interface after the user clicks to enter the game. This interface is mainly used for game management. The main menu is shown in Fig. 3. It mainly includes the following aspects: a new game, which is to create a new game record and experience the whole story flow of the game from the beginning; Pause button to realize the game pause function; Game parameter setting, mainly setting external parameters such as music and sound effects; Exit the game and end the whole game process.

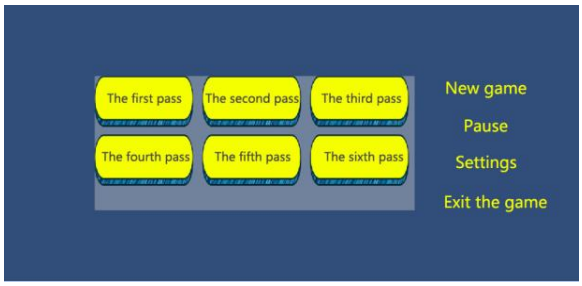


Fig. 3. Main menu

In-game interface: This interface is the interface after the player enters the game state (as shown in Fig. 4). In this interface, the player needs to carry out corresponding operations according to different game forms. The player needs to master the balance on the vine, and can choose to jump up

on the vine to collect gold coins, use the terrain to move forward and backward to avoid rolling stones, and get gold coins, avoid rolling stones, birds, etc. will get points, Each pass will be animated (as shown in Fig. 5). The difficulty of the game will increase gradually. The terrain of each pass is different. The terrain of the sixth pass is the most complex; Game Win animation will be triggered after winning all levels (as shown in Fi. 5); If it fails, the Game Over animation will be triggered (as shown in Fig. 5).

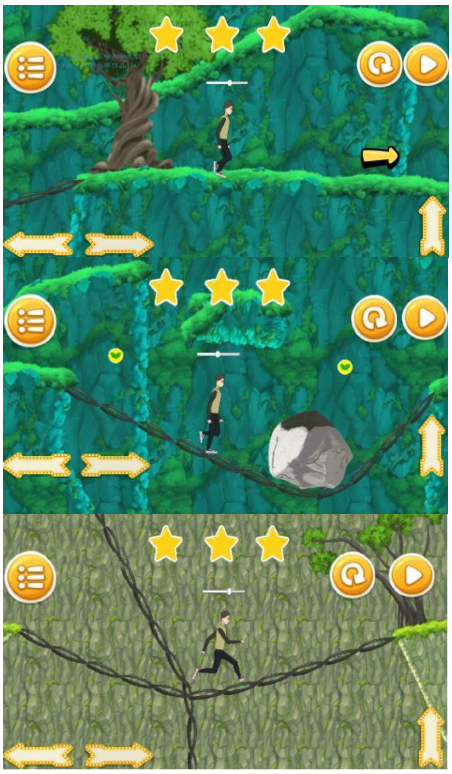


Fig. 4. Screenshot in the game

Fig. 5. Stage clearance, game clearance and failure interface

C. Input and Output Analysis

Input: Players can control the game through the shortcut keys on the screen. For example, the "Start" button starts the game, the "Pause" button pauses the game, the "Main Menu" button returns to the main menu, the "Volume" button controls the volume, and the "→", "←" and "↑" buttons control the forward, backward and jump of the game characters respectively.

Output: player's score. When the game character controlled by the player dies, a message "Game Over!" is output, indicating the death of the game character.

D. Key Technology Realization

We fill in a "Player" script for the protagonist. When the protagonist stands on the vine, the balance parameters of the protagonist will change. If the ball representing the balance moves to the leftmost or rightmost protagonist, it will fall, and the game will fail; All the "Ugui" in the game are integrated into the "object" of "HUD". The code "HUD" is used to select next level, volume control, advertising function, etc. The advertising function is written in the "HUD", and the relevant "Button" is hidden; We also created an "object" to generate a "flying bird". When the flying bird meets the protagonist on the vine, the protagonist will fall.

In the early stage of game design, it is planned to use the 2d script in the "Standard Asset" project in the "Asset Store" to realize the control of characters and the game control of Android platform. However, after changing the map of the 2d protagonist, the protagonist can't detect whether it touches the ground normally, so it can't control the jumping of the character. So we rewrite the script "PlatformerCharacter2D". We added a "gameobject" at the bottom of the main character to act as an object to detect whether it touches the ground, and continued to use "[SerializeField] private LayerMask m_WhatIsGround;". When the game was initially tested on the PC side, our team used "if (m_Grounded&&Input. GetKeyDown (KeyCode. Space)". But after it was released to the Android platform, we set a "Button" named "Jump" in the "object" of "Ugui", When you click "Jump", the method "jumpstart" will detect whether it touches the ground. If it touches the ground, the character can jump. The code is as follows:

```
m_Grounded = Physics2D.OverlapCircle(m_GroundCheck.position, k_GroundedRadius, m_WhatIsGround);
    m_Anim.SetBool("Ground", m_Grounded);
        void Jumpstart()
        {
            // if (m_Grounded && Input.GetKeyDown(KeyCode.Space))
            if (m_Grounded)
            {
                m_Grounded = false;
                int random1 = UnityEngine.Random.Range(-20, 20);
                Player.balance += random1;
                m_Rigidbody2D.AddForce(new Vector2(0, m_JumpForce));
            }
        }
```

However, the result is not ideal. By using "debug" to change the position of "groundcheck" at the foot of the protagonist, we find that "m_Grounded=Physics2D.

OverlapCircle (m_GroundCheck. position, k_GroundedRadius, m_WhatIsGround);" this detection method is effective. So we thought of putting "m_Anim. SetBool (" Ground ", m_Grounded)" into the "void Jumpstart()" method. After the jump, the animation state opportunity of the character is always triggered, and "m_Ground" is always set to "true", which has been verified by the "debug" method. Later, we used "if (m_Grounded&&Input. GetButtonDown (" Jump ")" to control the jump of the character. We found that the character cannot jump. We used "if (Input. GetButtonDown (" Jump ")) {Debug. log (" This is work ");}" to detect the problem, and found that it was "Input. GetButtonDown". Finally, we adjusted the animation playback time of the animation state machine to make the jump play as natural as possible.

E. Precautions for Game Operation

- When walking on the vine, pay attention to whether the balance bar on the top of the head is in the center to avoid falling off the vine.
- Pay attention to avoid the attack of rolling stones and birds.
- Pay attention to the change of terrain when evading the attack to avoid sudden falling into the air.

IV. GAME TEST

The testing principle is to try to comprehensively test and calmly deal with various problems encountered. The test strategy adopts the integrated test method, main test functions and performance. The test method adopts the white-box test method for each module at the early stage of design and the black-box test method at the later stage of game design.

Test whether the login is successful: test whether the tourists log in, click to enter the game, log in with a third-party account or mobile phone number, and then enter the game nickname setting after successful login. Select the game initial setting from the game option setting, and enter the main interface after the setting is completed, that is, login is successful.

Test whether the game can be played: after entering the main interface, select the character and select the tube card and the game map, automatically jump to the game interface and start the game, so that the game can be played successfully, otherwise, automatically exit the login interface, and the game fails.

Test whether the game saving operation can be carried out: automatically save after the game is over, or click the "Save" button to exit the game after saving successfully. The next time you log in to the game, you should read the archive first. If the reading is successful, you can continue the game by inheriting the ending position of the last game. If the reading fails, you can restart the game.

V. CONCLUSION

The game is designed with the purpose of convenience, practicality and high entertainment. In the process of interface design, it always adheres to clarity, and can achieve high efficiency and error-free in performance. The main interface of the game should be beautiful and pleasing to the eye. The game control module is easy to understand and operate, and has high accuracy and is not easy to make mistakes. Because the game is easy to operate and interactive, it has no special requirements for users. General users can be familiar with the rules of the game after a few minutes of practice. However, there are also some deviations between the actual development of this game and the expectation. The following two points are summarized for future research:

The design and function implementation of the game scene need to be improved. Only some relatively simple interactions have been realized, but a more humanized game experience has not been achieved. The focus of future work is to strengthen the construction of game scenes, increase the functions of games, and enhance the development technology of Unity 3D.

The game is only operated on Android phones and is a single player version. We will make some attempts on other systems in the future to reduce the limitations of the use area.

ACKNOWLEDGMENT

This paper is supported by the undergraduate teaching reform research project of Shandong Province "Research and Practice on the Integration of Ideological and Political Education and Innovation and Entrepreneurship Education" (M2021262) and the scientific research project of Shandong University of Political Science and Law (Innovation and Entrepreneurship Project) "Research on the Implementation Path of Ideological and Political Integration Course of Innovation and Entrepreneurship Education in Colleges and Universities" (2022CQ02Z).

REFERENCES

[1] Xu Xiuping, “Research on the development of children's puzzle games based on Android mobile platform”, Shanxi Youth, 2021 (02): 166-167.

[2] Xiao Fenglong, “Mobile game design research and analysis under mobile platform. Research and exchange”, 2019 (03): 260.

[3] Li Mengting, “ Mobile game design research and analysis on mobile platform”, Computer products and circulation, 2019 (02): 155.

[4] Fu Zhijie, “Design and implementation of ‘Three Kingdoms’ mobile game based on Android platform”, Fujian Computer, 2017 (09): 126-127.

[5] Wang Yinyin, Independent game development: foundation, practice and income generation (Unity 2D Android), China Machine Press, Beijing, 2020.

[6] Janarthanan Balakrishnan, Mark D. Griffiths, “Perceived Addictiveness of Smartphone Games: A Content Analysis of Game Reviews by Players”, International Journal of Mental Health and Addiction, Vol. 17, Issue 4, 2019, pp. 922-934.

山东师范大学图书馆检索证明报告 20231665

论文收录引用

检索报告

（检索日期：2023-09-01，检索责任人：李亚设）

委 托 人： 刘丽彦

委托人单位： 山东政法学院

发表论文单位： Shandong University of Political Science and Law, JiNan, China

检索要求： 刘丽彦发表的论文被 EI 收录

检索时段： 2023 年至今

检索结果

数据库	论文收录
EI Compendex	1 篇

声明：本次检索根据刘丽彦提供的论文目录，并按其提出的要求进行检索，详情见附件，若有不实，由委托人承担全部责任。

山东师范大学图书馆

（盖章）

二〇二三年九月四日

250014 山东师范大学图书馆
Tel: 0531-86180761 或 86181968
Email:375360605@qq.com

山东师范大学图书馆检索证明报告——附件

《工程索引》(EI)收录1篇

<RECORD 1>

Accession number:20232814384667
Title:Design and Implementation of Mobile Balance Games Based on Mobile Platforms
Authors:Liu, Liyan (1)
Author affiliation:(1) Shandong University of Political Science and Law, JiNan, China
Corresponding author:Liu, Liyan(Liuliyanid@163.com)
Source title:2023 4th Information Communication Technologies Conference, ICTC 2023
Abbreviated source title:Inf. Commun. Technol. Conf., ICTC
Part number:1 of 1
Issue title:2023 4th Information Communication Technologies Conference, ICTC 2023
Issue date:2023
Publication year:2023
Pages:285-288
Language:English
ISBN-13:9781665462587
Document type:Conference article (CA)
Conference name:4th Information Communication Technologies Conference, ICTC 2023
Conference date:May 17, 2023 - May 19, 2023
Conference location:Nanjing, China
Conference code:189980
Sponsor:IEEE; Nanjing University; Southeast University
Publisher:Institute of Electrical and Electronics Engineers Inc.
DOI:10.1109/ICTC57116.2023.10154756
Funding details: Number: M2021262, Acronym: -, Sponsor: -;Number: 2022CQ022, Acronym: SHUPL, Sponsor: Shanghai University of Political Science and Law;
Funding text:ACKNOWLEDGMENT This paper is supported by the undergraduate teaching reform research project of Shandong Province "Research and Practice on the Integration of Ideological and Political Education and Innovation and Entrepreneurship Education" (M2021262) and the scientific research project of Shandong University of Political Science and Law (Innovation and Entrepreneurship Project) "Research on the Implementation Path of Ideological and Political Integration Course of Innovation and Entrepreneurship Education in Colleges and Universities" (2022CQ007).
Database:Compendex
Compilation and indexing terms, Copyright 2023 Elsevier Inc.
委托人贡献:首作者、通讯作者

The Journal of Supercomputing (2019) 75:3099–3122
https://doi.org/10.1007/s11227-018-2534-z

A countermeasure against cryptographic key leakage in cloud: public-key encryption with continuous leakage and tampering resilience

Chengyu Hu[1,2] · Rupeng Yang[3] · Pengtao Liu[4] · Tong Li[5] · Fanyu Kong[6]

Published online: 23 August 2018

Abstract
Public-key encryption is an important security mechanism used in cloud environment. To ensure the confidentiality of data encrypted using public-key encryption, countermeasures against cryptographic key leakage by side-channel attacks should be applied to the encryption scheme implemented both in locality and in cloud server. Traditional public-key encryption does not capture side-channel attacks. Moreover, the adversary can inject fault to tamper with the secret key and observe the output of the public-key encryption scheme under this modified key which is called "tampering attack". In this paper, we present two continuous leakage and tampering resilient CCA secure public-key encryption schemes. For implementations of our schemes during the key update, bounded number of tampering queries for arbitrary key relations and bounded leakage is allowed. By updating the secret key, our schemes are secure against continuous leakage and tampering attacks.

Keywords Public-key encryption · Side-channel attacks · Tampering resilience · Leakage resilience · Continuous attacks

1 Introduction

Cloud computing provides a lot of services at very low cost so that there has been a shift toward IT applications implemented in cloud [1–3] which may face too many attacks [4]. For the data security in the cloud, rigorous security mechanisms are supplied by the cloud system. Firstly, in order to allow valid users to access what they are authorized, the cloud server will implement user authentication [5,6] and access control [7,8]. Secondly, for the data confidentiality, data owners encrypt the sensitive data before outsourcing it to the cloud which may brings challenges in data utilization, including machine learning over encrypted data [9,10], deduplication over ciphertext [11,12], keyword-based search on encrypted data [13–16], securely remote

Extended author information available on the last page of the article

operations [17,18]. As the data are stored in the remote server, there should be methods for the data owner checking data integrity [19,20] efficiently. In addition, some advance security, such as user's access pattern [21,22], should be considered during sharing and outsourcing encrypted data in the cloud [23–25].

The security of cryptographic keys plays an important role in the effect of cryptographic algorithms and protocols in the mechanism above, which can be broken by a kind of attacks called "side-channel attacks" [26–29]. Specifically for public-key encryption schemes used in cloud computing, as the decryption algorithm is usually executed by a client under malicious attacks, side-channel attacks can help recovering the decryption key to break the data confidentiality. Even though the decryption algorithm is executed by a trustworthy cloud server in the scenario that the cloud server may provide services to different clients by executing virtual machines (VMs) in the same physical machine and isolate each virtual machine to protect the privacy of each client, "cross-VM side-channel attacks" can help an adversary to extract some information about the decryption key stored in another VM which is carefully isolated [30–32]. That is to say, the secret decryption key may be leaked partially in a typical cloud setting which may break the cloud security. Also, the adversary can inject fault to tamper with the decryption key and observe the output of the decryption algorithm under this modified key which is called "tampering attacks" [33–35].

To protect against side-channel attacks and tampering attacks, ad-hoc approaches such as masking [36,37] can be applied to the corresponding algorithms. However, these approaches are sometimes rather expensive, inefficient and even impossible in some case which leads to the other approach using abstract notions of computation to model kinds of attacks and constructing schemes proved secure in the new model. Recently, the security models capturing side-channel attacks and tampering attacks were formalized, respectively, and many primitives secure in these models were proposed[33–35,38–44].

1.1 Related works

Continuous key-leakage attacks To capture more realistic side-channel attacks, the *continual memory leakage* (CML) model was proposed by [45,46], respectively. More precisely, the internal secret state of the scheme can be updated to a re-randomized one [47,48]. The adversary is allowed to obtain bounded leakage about the entire internal secret state between updates, but the total leakage over the lifetime of the scheme is unbounded. Several works gave concrete constructions of public-key encryption in this model [45,49–51].

Continuous key-leakage and tampering attacks Some works studied the schemes resilient to both key-leakage attacks and tampering attacks. In [52], the model covering both attacks called *continuous leakage and tampering* (CLT) model was considered and first feasibility results in this model were provided. As the CML model, the entire lifetime of the scheme is partitioned into some periods in this model. It allows the adversary to obtain bounded leakage information and arbitrarily tamper with the secret state and achieves continuous leakage and tampering resilient by updating the secret state at the end of each period. However, in their model the adversary cannot get the result

of decrypting a ciphertext using a given tampered key which weaken the adversary's ability, i.e., the scheme in [52] is IND-CPA (*Indistinguishable under Chosen-Plaintext Attacks*) secure rather than IND-CCA (*Indistinguishable under Chosen-Ciphertext Attacks*) secure. Moreover, the scheme in [52] only encrypts 1-bit plaintext once which is far from practical. In [53], the authors first constructed a scheme secure in the *bounded leakage and tamper* (BLT) model which allows the adversary to obtain bounded leakage and arbitrarily tamper with the secret state and then achieved continuous leakage and tampering resilience in the *floppy* model where the secret key can be refreshed using a securely preserved update key [54].

Our contribution In this paper, we focus on how to construct an efficient IND-CCA secure public-key encryption scheme with continuous leakage and tampering resilience. We summarize our contribution in the following:

- Give a somewhat inefficient IND-CCA secure public-key encryption scheme in CLT model. Specifically, we prove that the public-key encryption scheme in section 6.3.3 of Wichs's Phd thesis [55] is IND-CPA secure in CLT model which can be transformed to an inefficient IND-CCA secure public-key encryption scheme in CLT model using the *Naor-Yung double encryption paradigm* [56].
- Provide an efficient IND-CCA secure public-key encryption scheme in a weak CLT model which restricts the adversary to ask for less tampering queries. To this end, we extend Wichs's IND-CPA secure scheme to an efficient IND-CCA secure public-key encryption scheme in CML model. Then it is proved that the extended scheme is also IND-CCA secure in CLT model. To efficiently achieve IND-CCA security in CML/CLT model, we modify the original scheme of Wichs by introducing one-time lossy filter presented by [57]. Informally speaking, a one-time lossy filter is a family of functions indexed by a public key F_{pk} and a tag t which will be injective unless the tag comes from some specific set. Moreover, it is hard to generate and recognize such non-injective tags without a trapdoor F_{td} associated with F_{pk}. In our new scheme, F_{pk} is added to the public key, and we will apply $LF_{F_{pk},t}(\cdot)$ to the encapsulated key of the original scheme to verify the validity of the ciphertext where t is a random tag here. Since it is hard for the adversary to generate a non-injective tag, the output of $LF_{F_{pk},t}(\cdot)$ preserves the min-entropy of its input. Therefore, it is sufficient to prove that the min-entropy of the encapsulated key is high when the ciphertext is "bad-formed". However, the proof in [57] does not work in our setting. To see this, note that they can only prove that for any "bad ciphertext" the encapsulated key has high min-entropy with overwhelming probability while we need to prove that the encapsulated keys of all"bad ciphertexts" have high min-entropy with overwhelming probability. Therefore, we employ the proof technique in [58] to bridge the gap. Since one-time lossy filter is much more efficient compared to Wichs's public-key encryption scheme, our scheme is as efficient as the original scheme of Wichs.
- Unlike [53], our two schemes do not need a securely kept update key to refresh the decryption key.

1.2 Organization

In Sect. 2, we review some preliminaries. In Sect. 3, we show how to construct IND-CCA secure public-key encryption scheme in CLT model. We give the definition of CLT model. Also, we present an inefficient IND-CCA secure public-key encryption scheme in CLT model with linear tampering times and an efficient IND-CCA secure public-key encryption scheme in CLT model which can resist less tampering queries. Finally, we draw our conclusions in Sect. 4.

2 Preliminaries

In this section, we recall some basic notions, terminology and computational assumption.

2.1 Basic notions

We denote by U_m a random variable with uniform distribution over $\{0, 1\}^m$ and write $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

2.2 Linear Algebra

Let q be a prime, we make extensive use of linear algebra over $\mathbb{Z}_q$ in this paper. We use bold uppercase letters ($\mathbf{X}$) to denote matrices and lowercase letters with arrow ($\boldsymbol{x}$) to denote vectors. All vectors used in this paper are column vectors. Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ be m vectors in $\mathbb{Z}_q^n$, then we denote by $span(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ the linear space spanned by these vectors. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we denote by $colspan(\mathbf{A})$ and $rowspan(\mathbf{A})$ the space spanned by the column vectors and row vectors of $\mathbf{A}$, respectively. If $\mathcal{V}$ is a subspace of $\mathbb{Z}_q^{n\times m}$ with dimension $d < n$, we let $\mathcal{V}^\perp$ denote the orthogonal space of $\mathcal{V}$, which is defined as $\mathcal{V}^\perp = \{\boldsymbol{w} \in \mathbb{Z}_q^n \mid \forall \boldsymbol{v} \in \mathcal{V}, \boldsymbol{w} \cdot \boldsymbol{v} = 0\}$. It is easy to see that the dimension of $\mathcal{V}^\perp$ is $n - d$. We use $Rk_d(\mathbb{Z}_q^{n\times k})$ to denote the set of $n \times k$-matrices with rank d. We also denote by $Rk_d(\mathbb{Z}_q^{n\times k}) \mid col \in \mathcal{W})$ the set of rank d matrices in $\mathbb{Z}_q^{n\times k}$ whose columns come from $\mathcal{W}$ and write $U(\mathbb{Z}_q^{n\times k} \mid col \in \mathcal{W})$ as abbreviation for $U(\{[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k] \mid \boldsymbol{a}_i \in \mathcal{W}, i = 1, \ldots, k\})$ where $\mathcal{W}$ is a subspace of $\mathbb{Z}_q^n$.

Random subspaces are leakage resilient One crucial property of random subspaces we exploit to resist continuous memory attacks is that they are leakage resilient. More precisely, it is hard to distinguish whether a set of vectors are chosen uniformly at random from a larger space or a smaller subspace of the former space as long as the distinguisher can only obtain bounded information about these vectors. This property was first formulated by [45] and improved by Wichs latter in his Phd thesis [55].

Lemma 1 *Let $n \geq d \geq u$ be positive integers. Let $f : \{0, 1\}^* \to \{0, 1\}^l$ be some arbitrary function. For randomly sampled $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times d}$, $\mathbf{V} \leftarrow \mathbb{Z}_q^{d\times u}$, $\mathbf{U} \leftarrow \mathbb{Z}_q^{n\times u}$, we have:*

$$(f(\mathbf{A} \cdot \mathbf{V}), \mathbf{A}) \approx_{negl(\lambda)} (f(\mathbf{U}), \mathbf{A})$$

as long as $(d-u)log(q)-l = \omega(log(\lambda))$,$n = poly(\lambda)$ *and* $q = \lambda^{\omega(1)}$ *where* λ *is the security parameter.*

Note that, the function f must be independent of $\mathbf{A}$. Therefore, information about $\mathbf{A}$ must not be revealed until f has been chosen.

Linear map A *basis* of a subspace $\mathcal{V} \subseteq \mathbb{Z}_q^n$ is a set of linearly independent vectors $\boldsymbol{v}_1, \cdot, \boldsymbol{v}_m \in \mathcal{V}$ that can span $\mathcal{V}$. Let $\mathbf{B} = [\boldsymbol{v}_1, \cdot, \boldsymbol{v}_m]$ and consider the map $f(\mathbf{X}) = \mathbf{B} \cdot \mathbf{X}$ for $\mathbf{X} \in \mathbb{Z}_q^{m \times l}$. It is obvious that f is bijective from $\mathbb{Z}_q^{m \times l}$ to $\mathcal{V}^l$. Also, since $\mathbf{B}$ has full column rank, the rank of $f(\mathbf{X})$ is equal to the rank of $\mathbf{X}$. In particular, if $\mathbf{X}$ is chosen from $Rk_m(\mathbb{Z}_q^{m \times m})$, the result of $f(\mathbf{X})$) is also a matrix whose columns form a basis of $\mathcal{V}$.

When a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ has full row rank, the map $f(\mathbf{X}) = \mathbf{A} \cdot \mathbf{X}$ is surjective as claimed in the following lemma which is a variant of lemma 9 in [59].

Lemma 2 *For any positive integers* m, n, l *s.t.* $m \leq n$, *for any matrix* $\mathbf{A} \in Rk_m(\mathbb{Z}_q^{m \times n})$, $f(\mathbf{X}) = \mathbf{A} \cdot \mathbf{X}$ *is a surjective map from* $\mathbb{Z}_q^{n \times l}$ *to* $\mathbb{Z}_q^{m \times l}$.

The following corollary which indicates the distribution of $f(\mathbf{X})$ when $\mathbf{X}$ is distributed uniformly in $\mathbb{Z}_q^{n \times l}$ follows directly from Lemma 2 and the homomorphism property of a linear map.

Corollary 1 *For any positive integers* m, n, l *s.t.* $m \leq n$, *for any matrix* $\mathbf{A} \in Rk_m(\mathbb{Z}_q^{m \times n})$, *let* $f(\mathbf{X}) = \mathbf{A} \cdot \mathbf{X}$ *be a map from* $\mathbb{Z}_q^{n \times l}$ *to* $\mathbb{Z}_q^{m \times l}$. *If* $\mathbf{X}$ *is chosen uniformly at random from* $\mathbb{Z}_q^{n \times l}$, *then* $f(\mathbf{X})$ *is distributed uniformly over* $\mathbb{Z}_q^{m \times l}$.

Vector sampling In some cases, the manner of sampling vectors may have only a minimal effect.

Lemma 3 [45] *For any positive integers* m, n *s.t.* $m \leq n$, $SD(U(\mathbb{Z}_q^{n \times m}), U(Rk_m(\mathbb{Z}_q^{n \times m}))) \leq \frac{1}{q^{n-m}(q-1)}$.

Lemma 4 [55] *If* $\mathcal{W} \subseteq \mathbb{Z}_q^n$ *is a fixed subspace of dimension* $\geq d$, *the uniform distribution over* $Rk_d(\mathbb{Z}_q^{n \times m} \mid col \in \mathcal{W})$ *is equivalent to sampling* $\mathbf{C} \leftarrow Rk_d(\mathbb{Z}_q^{n \times m} \mid col \in \mathcal{W})$, $\mathbf{R} \leftarrow Rk_d(\mathbb{Z}_q^{d \times m})$ *and outputting* $\mathbf{A} = \mathbf{C} \cdot \mathbf{R}$.

Lemma 5 *For any positive integers* m, n, l, i, j *s.t.* $i \leq m$, $j \leq n$ *and* $i + j \leq l$, *let* $\mathbf{X}$ *be a random variable with distribution* $U(Rk_i(\mathbb{Z}_q^{l \times m}))$, $\mathbf{Y}$ *be a random variable with distribution* $U(Rk_j(\mathbb{Z}_q^{l \times n} \mid col \in colspan(\mathbf{X})^{\perp}))$, $\mathbf{V}$ *be a random variable with distribution* $U(Rk_j(\mathbb{Z}_q^{l \times n}))$, *and* $\mathbf{W}$ *be a random variable with distribution* $U(Rk_i(\mathbb{Z}_q^{l \times m} \mid col \in colspan(\mathbf{V})^{\perp}))$. *Then* $SD((\mathbf{X}, \mathbf{Y}), (\mathbf{W}, \mathbf{V})) = 0$.

Proof By Lemma 4, we can write $\mathbf{X} = \mathbf{X}' \cdot \mathbf{R}_1$, $\mathbf{Y} = \mathbf{Y}' \cdot \mathbf{R}_2$, $\mathbf{W} = \mathbf{W}' \cdot \mathbf{R}_3$ and $\mathbf{V} = \mathbf{V}' \cdot \mathbf{R}_4$ where $\mathbf{X}', \mathbf{Y}', \mathbf{V}', \mathbf{W}'$ are random variables with distributions $U(Rk_i(\mathbb{Z}_q^{l \times i}))$, $U(Rk_j(\mathbb{Z}_q^{l \times j} \mid col \in colspan(\mathbf{X}')^{\perp}))$, $U(Rk_j(\mathbb{Z}_q^{l \times j}))$, $U(Rk_i(\mathbb{Z}_q^{l \times i} \mid col \in colspan(\mathbf{V}')^{\perp}))$, respectively. And $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \mathbf{R}_4$ are chosen uniformly and independently at random from $Rk_i(\mathbb{Z}_q^{i \times m})$, $Rk_j(\mathbb{Z}_q^{j \times n})$, $Rk_i(\mathbb{Z}_q^{i \times m})$, $Rk_j(\mathbb{Z}_q^{j \times n})$, respectively. Therefore, it is suffice to prove $SD((\mathbf{X}', \mathbf{Y}'), (\mathbf{W}', \mathbf{V}')) = 0$.

For any tuple $(\mathbf{A}, \mathbf{B}) \in Rk_i(\mathbb{Z}_q^{l\times i}) \times Rk_j(\mathbb{Z}_q^{l\times j})$ s.t. $\mathbf{A}^T \cdot \mathbf{B} = 0$, we have

$$\begin{aligned}&\Pr[(\mathbf{X}', \mathbf{Y}') = (\mathbf{A}, \mathbf{B})]\\ &= \frac{1}{(q^l-1)\cdots(q^l-q^{i-l})} \cdot \frac{1}{(q^{l-i}-1)\cdots(q^{l-i}-q^{j-l})}\\ &= \frac{1}{(q^l-1)\cdots(q^{l-i-j+1}-1)\cdot q^{i(i-1)/2+j(j-1)/2}}\end{aligned}$$

and

$$\begin{aligned}&\Pr[(\mathbf{W}', \mathbf{V}') = (\mathbf{A}, \mathbf{B})]\\ &= \frac{1}{(q^l-1)\cdots(q^l-q^{j-l})} \cdot \frac{1}{(q^{l-j}-1)\cdots(q^{l-j}-q^{i-l})}\\ &= \frac{1}{(q^l-1)\cdots(q^{l-i-j+1}-1)\cdot q^{i(i-1)/2+j(j-1)/2}}\end{aligned}$$

Further, for any tuple $(\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{l\times i} \times \mathbb{Z}_q^{l\times j}$ that does not satisfy the above condition, we have $\Pr[(\mathbf{X}', \mathbf{Y}') = (\mathbf{A}, \mathbf{B})] = \Pr[(\mathbf{W}', \mathbf{V}') = (\mathbf{A}, \mathbf{B})] = 0$. Combining the two cases, the lemma follows. □

We also have the following corollary directly from Lemmas 3 and 5.

Corollary 2 *For any positive integers m, n, l s.t. $m + n \le l$, let $\mathbf{X}$ be a random variable with distribution $U(\mathbb{Z}_q^{l\times m})$, $\mathbf{Y}$ be a random variable with distribution $U(\mathbb{Z}_q^{l\times n} \mid col \in colspan(\mathbf{X})^{\perp})$, $\mathbf{V}$ be a random variable with distribution $U(\mathbb{Z}_q^{l\times n})$, $\mathbf{W}$ be a random variable with distribution $U(\mathbb{Z}_q^{l\times m} \mid col \in colspan(\mathbf{V})^{\perp})$. Then $SD((\mathbf{X}, \mathbf{Y}), (\mathbf{W}, \mathbf{V})) \le \frac{4}{q^{l-m-n}(q-1)}$.*

2.3 Randomness extractor

We recall some basic notions relating to randomness extractors here.

Definition 1 Let X and Y be two random variables in a finite set U. The statistical distance between X and Y is defined as

$$\mathrm{SD(X,Y)} = \frac{1}{2}\sum_{u\in U} |Pr[X = u] - Pr[Y = u]|$$

We say that two variables are $\epsilon-close$ if their statistical distance is at most ϵ.

Definition 2 [60] Let X be a random variable. Then the min-entropy of X, denoted as $H_\infty(X)$, is defined as

$$H_\infty(X) = -log(max_x Pr[X = x])$$

Definition 3 [60] Let X and Y be two random variables. Then the average min-entropy of X conditioned on Y, denoted as $\widetilde{H}_\infty(X|Y)$, is defined as

$$\begin{aligned}\widetilde{H}_\infty(X|Y) &= -log(\mathbb{E}_{y\leftarrow Y}[max_{x\leftarrow X}Pr[X=x|Y=y]]) \\ &= -log(\mathbb{E}_{y\leftarrow Y}[2^{-H_\infty(X|Y=y)}])\end{aligned}$$

Lemma 6 *Let X, Y and Z be random variables. If Y has at most* 2^λ *possible values, then* $\widetilde{H}_\infty(X|(Y,Z)) \geq \widetilde{H}_\infty((X,Y)|Z) - \lambda \geq \widetilde{H}_\infty(X|Z) - \lambda$.

The proof of this lemma can be found in [60].

We will also consider the decrease in "worst case" min-entropy when given some bounded bits of information.

Lemma 7 ([50]) *Let X be a random variable with min-entropy h and let f be an arbitrary function with range* $\{0,1\}^l$. *For any* $\tau \in [0, h-l]$, *we define the set*

$$V_\tau = \{v \in \{0,1\}^l \mid H_\infty(X \mid f(X)=v) \leq h - l - \tau\}$$

Then:

$$Pr[f(X) \in V_\tau] \leq 2^{-\tau}$$

Definition 4 Let U be a finite set. A function Ext: $U \times \{0,1\}^t \rightarrow \{0,1\}^m$ is an average-case (v,ϵ)-strong extractor if for all pairs of random variables (X, I) such that the range of X is U and $\widetilde{H}_\infty(X|I) \geq v$, it holds that

$$\text{SD}((\text{Ext(X,R)},\text{R},\text{I}), (U_m, \text{R},\text{I})) \leq \epsilon$$

where R is uniform on $\{0,1\}^t$.

We remark that as we can easily encode elements in U into binary strings in our settings, we just write *Ext* as a function from $U \times \{0,1\}^t$ to $\{0,1\}^m$ for the given U for simplicity.

2.4 Hardness Assumption in Bilinear Groups

Our scheme works in the bilinear groups which are defined as below. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three groups of prime order q, and g, h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficient map s.t.:

1. Bilinearity: for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ where $1_{\mathbb{G}_T}$ is the identity element in $\mathbb{G}_T$.

We will rely on the symmetric *external Diffie–Hellman* (SXDH) assumption [61] in bilinear groups as above. The SXDH assumption implies that the *decisional Diffie–Hellman* (DDH) problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. However, we will actually use

the following *d-rank hiding* assumption introduced by [58] in each group which will hold as long as the DDH problem is hard in respective group.

***d*-Rank hiding** Let $\mathrm{Rk}_d(\mathbb{Z}_q^{n\times m})$ denote the set of all $n \times m$ matrices over $\mathbb{Z}_q$ with rank d. The d-rank hiding assumption [58] on G which was shown to be implied by the d-linear assumption states that for any constants $d \leq i < j \leq \min\{m, n\}$, we cannot distinguish rank i and j matrices in the exponent of g: $(\mathbb{G}, g, g^X) \approx_c (\mathbb{G}, g, g^Y)$ where $X \leftarrow \mathrm{Rk}_i(\mathbb{Z}_q^{n\times m})$, $Y \leftarrow \mathrm{Rk}_j(\mathbb{Z}_q^{n\times m})$.

Also, we will work with matrices and vectors in above groups. Let $\mathbf{R} = \{r_{i,j}\}_{i\in[m],j\in[n]}$ be a matrix in $\mathbb{Z}_q^{n\times m}$, and we denote by $g^{\mathbf{R}}$ the matrix $\{g_{i,j}\}_{i\in[m],j\in[n]} \in \mathbb{G}_1^{m\times n}$. We can also define matrices in $\mathbb{G}_2$ and $\mathbb{G}_T$ analogously. Let $\boldsymbol{a}, \boldsymbol{b}$ be two vectors in $\mathbb{Z}_q^n$, and we define $e(g^{\boldsymbol{a}}, h^{\boldsymbol{b}}) = e(g, h)^{\boldsymbol{a}\cdot\boldsymbol{b}}$. This can be computed easily since

$$e(g,h)^{\boldsymbol{a}\cdot\boldsymbol{b}} = e(g,h)^{\sum_{i=1}^n a_i b_i} = \prod_{i=1}^n e(g,h)^{a_i b_i} = \prod_{i=1}^n e(g^{a_i}, h^{b_i}).$$

2.5 One-time lossy filter [57]

A (Dom,l_{LF})-OT-LF is a family of functions indexed by a public key F_{pk} and a tag t. A function $\mathrm{LF}_{F_{pk},t}$ from the family maps an input $X \in$Dom to an output $\mathrm{LF}_{F_{pk},t}(X)$. Given public key F_{pk}, the set of tags $\mathcal{T}$ contains two computationally indistinguishable disjoint subsets, namely the subset of injective tags $\mathcal{T}_{inj}$ and the subset of lossy ones $\mathcal{T}_{loss}$. If t is an injective tag, the function $\mathrm{LF}_{F_{pk},t}$ is injective and has image size of |Dom|. If t is lossy, the output of the function has image size at most $2^{l_{\mathrm{LF}}}$. Thus, a lossy tag ensures that $\mathrm{LF}_{F_{pk},t}(X)$ reveals at most $2^{l_{\mathrm{LF}}}$ bits of information about its input X. This is a crucial property of an LF. We will modify the definition of OT-LF a bit by adding an additional property for the sake of simplicity in our proof. Note that, this property can be achieved by making a slight modification to the original construction and we just omit the details here.

Definition 5 (*OT-LF*) A (Dom,l_{LF})-one-time lossy filter consists of three PPT algorithms (LF.Gen,LF.Eval,
LF.LTag,LF.DITag):

- **Key generation** LF.Gen(1^k) outputs a key pair (F_{pk}, F_{td}). The public key F_{pk} defines a tag space $\mathcal{T} = \{0,1\}^* \times \mathcal{T}_c$ that contains two disjoint subsets, the subset of lossy tags $\mathcal{T}_{loss} \subseteq \mathcal{T}$ and that of injective tags $\mathcal{T}_{inj} \subseteq \mathcal{T}$. A tag $t = (t_a, t_c) \in \mathcal{T}$ consists of an auxiliary tag $t_a \in \{0,1\}^*$ and a core tag $t_c \in \mathcal{T}_c$. F_{td} is a trapdoor that allows to efficiently sample a lossy tag.
- **Evaluation** LF.Eval(F_{pk}, t, X) for a public key F_{pk}, a tag t and $X \in$Dom, computes $\mathrm{LF}_{F_{pk},t}(X)$.
- **Lossy tag generation** LF.LTag(F_{td}, t_a), for an auxiliary tag t_a and the trapdoor F_{td}, computes a core tag t_c such that $t = (t_a, t_c)$ is lossy.
- **Injective tag decision** LF.DITag(F_{td}), for a tag t and the trapdoor F_{td}, decides whether t is an injective tag.

An OT-LF LF has the following properties:

– **Lossiness** If t is injective, so is the function $\mathrm{LF}_{F_{pk},t}(\cdot)$. If t is lossy, then $\mathrm{LF}_{F_{pk},t}(X)$ has image size of at most $2^{l_{LF}}$. (*In application, we are interested in* OT-LFs *that have a constant parameter* l_{LF} *even for larger domain.*)
– **Indistinguishability** For any PPT adversary $\mathcal{A}$, it is hard to distinguish a lossy tag from a random tag, i.e., the following advantage is negligible in k.

$$Adv^{ind}_{LF,\mathcal{A}}(k) := |\ \mathbf{Pr}[\mathcal{A}(F_{pk}, (t_a, t_c^{(0)})) = 1] - \mathbf{Pr}[\mathcal{A}(F_{pk}, (t_a, t_c^{(1)})) = 1\ |$$

where $(F_{pk}, F_{td}) \leftarrow LF.Gen(1^k)$, $t_a \leftarrow \mathcal{A}(F_{pk})$, $t_c^{(0)} \leftarrow LF.LTag(F_{td}, t_a)$ and $t_c^{(1)} \leftarrow \mathcal{T}_c$.
– **Evasiveness** For any PPT adversary $\mathcal{A}$, it is hard to generate a non-injective tag even given a lossy tag, i.e., the following advantage is negligible in k.

$$Adv^{eva}_{LF,\mathcal{A}}(k) := \mathbf{Pr}\left[\begin{array}{c} (t_a^{'}, t_c^{'}) \neq (t_a, t_c) \wedge \\ (t_a^{'}, t_c^{'}) \in \mathcal{T} \backslash \mathcal{T}_{inj} \end{array} : \begin{array}{c} (F_{pk}, F_{td}) \leftarrow LF.Gen(1^k); \\ t_a \leftarrow \mathcal{A}(F_{pk}); \\ t_c \leftarrow LF.LTag(F_{td}, t_a); \\ (t_a^{'}, t_c^{'}) \leftarrow \mathcal{A}(F_{pk}, (t_a, t_c)) \end{array}\right]$$

In [57], the authors give an efficient construction of OT-LF based on the standard DDH assumption.

3 Public-key encryption scheme with continuous leakage and tampering resilience

3.1 The model

To get a continuous leakage and tampering resilient public-key encryption scheme, we should equip the scheme with an update algorithm which can output a re-randomized key taking a secret key as input. Thus, a public-key encryption scheme with key update consists of four algorithms, KeyGen, Enc, Dec and KeyUpdate:

KeyGen(1^n):	It takes as input the security parameter 1^n and produces a public-key/private-key pair (pk, sk) from the key space $\mathcal{PK} \times \mathcal{SK}$.
Enc(pk, m):	It takes as input a public key pk and a message m from the message space $\mathcal{M}$ and outputs a ciphertext c.
Dec(sk, c):	It takes as input the secret key sk and a ciphertext c and outputs either a message $m' \in \mathcal{M}$ or a reject symbol $\perp$.
KeyUpdate(pk, sk):	It takes as input a secret key sk and outputs a re-randomized secret key sk' such that $\|sk\| = \|sk'\|$. The distribution of sk is indistinguishable from the distribution of sk'.

It is required that for all $m \in \mathcal{M}$ and $t \in \mathbb{N}$,

$$\mathbf{Pr}\left[\begin{array}{c} c = Enc(pk, m) \\ Dec(sk_t, c) \neq m \end{array} : \begin{array}{c} (pk, sk_0) \leftarrow KeyGen(1^n); \\ sk_i \leftarrow KeyUpdate(pk, sk_{i-1}), \forall i \in [t] \end{array}\right] \leq negl(n)$$

To construct a public-key encryption scheme with continuous leakage and tampering resilience, we first present an IND-CCA secure public-key encryption scheme with continuous leakage resilience and then prove its tampering resilience. Therefore, we need describe chosen-ciphertext security for public-key encryption scheme in both CML and CLT model in the following.

Definition 6 (*IND-CCA CML Security*) A public-key encryption scheme Γ=(KeyGen, Enc, Dec, KeyUpdate) is secure against chosen-ciphertext attacks in the CML model with leakage amount λ if for any PPT adversary $\mathcal{A}$ we have Pr[$\mathcal{A}$ wins]$\leq \frac{1}{2}$+negl(n) in the following game:

- Initialize The challenger generates $(pk, sk_0) \leftarrow \text{KeyGen}(1^n)$ and sends pk to $\mathcal{A}$. It also sets the current secret key $sk = sk_0$ and $L = 0$.
- Priori Queries $\mathcal{A}$ can adaptively make any number of the following queries:
  - **Leakage queries** $\mathcal{A}$ queries $\mathcal{O}(leak, f)$ where f is an efficient computable function. If $L+|f(sk)| \leq \lambda$, then the challenger returns $f(sk)$ to the adversary and sets $L = L + |f(sk)|$. Otherwise, the challenger aborts.
  - **Decryption queries** $\mathcal{A}$ queries $\mathcal{O}(dec, c)$. The challenger returns Dec(sk,c) to the adversary.
  - **Update queries** $\mathcal{A}$ queries $\mathcal{O}(update, \cdot)$. The challenger computes $sk' =$ KeyUpdate(sk) and then sets the current secret key $sk = sk'$ and $L = 0$.
- Challenge $\mathcal{A}$ sends two messages m_0^*, m_1^* to the challenger. The challenger chooses $b \in \{0, 1\}$, computes $c^* \leftarrow$Enc(pk, m_b^*) and sends c^* to $\mathcal{A}$.
- Posteriori Queries $\mathcal{A}$ can further adaptively make any number of the following queries:
  - **Decryption queries** $\mathcal{A}$ queries $\mathcal{O}(dec, c)$. If $c \neq c^*$, then the challenger returns Dec(sk,c) to the adversary. Otherwise, the challenger aborts.
  - **Update queries** $\mathcal{A}$ queries $\mathcal{O}(update, \cdot)$. The challenger computes $sk' =$ KeyUpdate(sk) and then sets the current secret key $sk = sk'$ and $L = 0$.
- Guess $\mathcal{A}$ output $b' \in \{0, 1\}$.

The adversary $\mathcal{A}$ wins the game if $b' = b$.

Definition 7 (*IND-CCA CLT Security*) Let Φ_{sk} be some set of functions such that $\phi \in \Phi_{sk}$ has a type $\phi : \mathcal{SK} \rightarrow \mathcal{SK}$. A public-key encryption scheme $\Gamma =$(KeyGen, Enc, Dec, KeyUpdate) is secure against chosen-ciphertext attacks in the CLT model with respect to Φ_{sk} if for any PPT adversary $\mathcal{A}$ we have Pr[$\mathcal{A}$ wins]$\leq \frac{1}{2}$+negl(n) in the following game:

- Initialize The challenger generates $(pk, sk_0) \leftarrow \text{KeyGen}(1^n)$ and sends pk to $\mathcal{A}$. It also sets the current secret key $sk = sk_0$ and $L = 0$, $T = 0$.
- Priori Queries $\mathcal{A}$ can adaptively make any number of the following queries:
  - **Leakage queries** $\mathcal{A}$ queries $\mathcal{O}(leak, f)$ where f is an efficient computable function. If $L+|f(sk)| \leq \lambda$, then the challenger returns $f(sk)$ to the adversary and sets $L = L + |f(sk)|$. Otherwise, the challenger aborts.

  - **Decryption queries** $\mathcal{A}$ queries $\mathcal{O}(dec, c)$. The challenger returns Dec(sk,c) to the adversary.
  - **Tampering queries** $\mathcal{A}$ queries $\mathcal{O}(tamper, \phi)$ where $\phi \in \Phi_{sk}$ is a tampering function. If $T + 1 > t$, the challenger aborts. Otherwise, the challenger sets $\widetilde{sk} = \phi(sk)$, $T = T + 1$. This oracle also outputs decryption of ciphertexts using $\widetilde{sk}$ when receiving polynomially many ciphertexts c chosen by the adversary itself, i.e., this oracle outputs polynomially many $\tilde{m}$=Dec$(\phi(sk), c)$.
  - **Update queries** $\mathcal{A}$ queries $\mathcal{O}(update, \cdot)$. The challenger computes $sk' =$ KeyUpdate(sk) and then sets the current secret key $sk = sk'$, $L = 0$ and $T = 0$.

- Challenge $\mathcal{A}$ sends two messages m_0^*, m_1^* to the challenger. The challenger chooses $b \in \{0, 1\}$, computes $c^* \leftarrow$Enc(pk, m_b^*) and sends c^* to $\mathcal{A}$.
- Posteriori Queries $\mathcal{A}$ can further adaptively make any number of the following queries:

  - **Decryption queries** $\mathcal{A}$ queries $\mathcal{O}(dec, c)$. If $c \neq c^*$, then the challenger returns Dec(sk,c) to the adversary. Otherwise, the challenger aborts.
  - **Update queries** $\mathcal{A}$ queries $\mathcal{O}(update, \cdot)$. The challenger computes $sk' =$ KeyUpdate(sk) and then sets the current secret key $sk = sk'$, $L = 0$ and $T = 0$.

- Guess $\mathcal{A}$ output $b' \in \{0, 1\}$.

The adversary $\mathcal{A}$ wins the game if $b' = b$.

To construct an IND-CCA CLT-secure public-key encryption scheme which can resist more tampering queries, we will first obtain an IND-CPA CLT-secure public-key encryption scheme and compile it to an IND-CCA CLT-secure public-key encryption scheme using the *Naor-Yung double encryption paradigm* [56]. Therefore, we will give the formal definition of IND-CPA CLT security of public-key encryption in the following. As same as that in [53], we only allow the adversary to get the output of the tampered decryption oracle for ciphertexts c which it knows both the corresponding plaintext m and the used randomness r.

Definition 8 (*IND-CPA CLT Security*) Let Φ_{sk} be some set of functions such that $\phi \in \Phi_{sk}$ has a type $\phi : \mathcal{SK} \rightarrow \mathcal{SK}$. A public-key encryption scheme Γ = (KeyGen, Enc, Dec, KeyUpdate) is secure against chosen-plaintext attacks in the CLT model with respect to Φ_{sk} if for any PPT adversary $\mathcal{A}$ we have Pr[$\mathcal{A}$ wins]$\leq \frac{1}{2}$+negl(n) in the following game:

- Initialize The challenger generates $(pk, sk_0) \leftarrow$KeyGen(1^n) and sends pk to $\mathcal{A}$. It also sets the current secret key $sk = sk_0$ and $L = 0$, $T = 0$.
- Queries $\mathcal{A}$ can adaptively make any number of the following queries:

  - **Leakage queries** $\mathcal{A}$ queries $\mathcal{O}(leak, f)$ where f is an efficient computable function. If $L + |f(sk)| \leq \lambda$, then the challenger returns $f(sk)$ to the adversary and sets $L = L + |f(sk)|$. Otherwise, the challenger aborts.
  - **Tampering queries** $\mathcal{A}$ queries $\mathcal{O}(tamper, \phi)$ where $\phi \in \Phi_{sk}$ is a tampering function. If $T + 1 > t$, the challenger aborts. Otherwise, the challenger sets

$\widetilde{sk} = \phi(sk)$, $T = T + 1$. When receiving (m, r) pairs chosen by the adversary itself, this oracle computes $c \leftarrow \text{Enc}(pk, m; r)$ and outputs decryption of ciphertexts c using $\widetilde{sk}$, i.e., $\tilde{m} \leftarrow \text{Dec}(\phi(sk), c)$.

- **Update queries** $\mathcal{A}$ queries $\mathcal{O}(update, \cdot)$. The challenger computes $sk' = \text{KeyUpdate}(sk)$ and then sets the current secret key $sk = sk'$, $L = 0$ and $T = 0$.

- Challenge $\mathcal{A}$ sends two messages m_0^*, m_1^* to the challenger. The challenger chooses $b \in \{0, 1\}$, computes $c^* \leftarrow \text{Enc}(pk, m_b^*)$ and sends c^* to $\mathcal{A}$.
- Guess $\mathcal{A}$ output $b' \in \{0, 1\}$.

The adversary $\mathcal{A}$ wins the game if $b' = b$.

3.2 The construction resisting more tampering queries and inefficiency

In this section, we present an inefficient public-key encryption scheme which can resist more tampering queries. We show that that the public-key encryption scheme in section 6.3.3 of [55] is IND-CPA CLT-secure which can resist tampering queries for linear times. Then we can transform it to an IND-CCA CLT-secure public-key encryption scheme using the *Naor-Yung double encryption paradigm* [56] which is inefficient since it applies non-interactive zero-knowledge proof systems to verify the validity of ciphertexts.

Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ is a pairing group of prime order q, where q is $poly(n)$-bits long and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $g, h, e(g, h)$ be generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively. Let $l \geq 5$ be some constant. The Wichs's scheme is parameterized by above parameters as well as the security parameter n and works as follows:

KeyGen(1^n):	The key generation algorithm samples $\boldsymbol{\alpha}, \boldsymbol{s}_0 \leftarrow \mathbb{Z}_q^l$ and $\boldsymbol{w} \leftarrow span(\boldsymbol{\alpha})^{\perp}$. Then it sets $PK = (g^{\boldsymbol{\alpha}}, e(g, h)^{\boldsymbol{\alpha} \cdot \boldsymbol{s}_0}, h^{\boldsymbol{w}})$ and $SK = h^{\boldsymbol{s}_0}$.
Enc(pk, M):	Given a public key $PK = (g^{\boldsymbol{\alpha}}, e(g, h)^{\boldsymbol{\alpha} \cdot \boldsymbol{s}_0}, h^{\boldsymbol{w}}, F_{pk})$, to encrypt a message $M \in \{0, 1\}^m$, the encryption algorithm samples $r \leftarrow \mathbb{Z}_q$ and computes $C = (g^{\boldsymbol{\alpha}})^r$, $K = (e(g, h)^{\boldsymbol{\alpha} \cdot \boldsymbol{s}_0})^r$, $Z = K \cdot M$. Then it outputs $CT = (C, Z)$ as ciphertext.
Dec(SK, CT):	Given a secret key $SK = h^s$, to decrypt a ciphertext $CT = (C, Z)$, the decryption algorithm computes $K' = e(C, SK)$ and outputs $M' = Z/K'$.
KeyUpdate(PK, SK):	Given a secret key $SK = h^s$ and a public key $PK = (g^{\boldsymbol{\alpha}}, e(g, h)^{\boldsymbol{\alpha} \cdot \boldsymbol{s}_0}, h^{\boldsymbol{w}})$, the update algorithm samples $t \leftarrow \mathbb{Z}_q$ and outputs a new key $SK' = h^s \cdot (h^{\boldsymbol{w}})^t$.

The above scheme is proved IND-CPA CML-secure with leakage resilient up to $\lambda = (l - 4)\log(q)$–$\omega(\log(n))$. In the following, we prove that the above scheme is IND-CPA CLT-secure. Informally speaking, we can simulate decryption queries for a given tampered key in the tampering queries oracle by only leaking a bounded amount of information from the secret key. Hence, security in CLT model follows from the security in CML model.

Theorem 3.1 *The above encryption scheme is (λ', t')-secure against chosen-plaintext attacks in the CLT model, where*

$$\lambda' \leq (l - 4 - t')log(q) - \omega(log(n)) \text{ and } t' \leq l - 5$$

Proof To prove this, assuming an adversary $\mathcal{A}$ can break IND-CPA CLT security of the scheme, then we construct an adversary $\mathcal{B}$ to break IND-CPA CML security of the scheme. Adversary $\mathcal{B}$ simulates environment for $\mathcal{A}$ and uses $\mathcal{A}$ as a black box as follows:

- $\mathcal{B}$ is given (PK, SK) from its challenger and forwards them to $\mathcal{A}$.
- *Leakage Queries*. Whenever $\mathcal{A}$ asks for a leakage query, $\mathcal{B}$ submits this query to its challenger and returns the answer directly to $\mathcal{A}$.
- *Tampering Queries*. Upon receiving a tampering query $\mathcal{O}(tamper, \phi_i)$ where $\phi_i \in \Phi_{sk}$ is a tampering function, $\mathcal{B}$ firstly checks whether $T + 1 \leq t'$ and returns the result by submitting a leakage function f to its own leakage oracle such that:

(1) Set $\widetilde{SK}_i = \phi_i(SK) = \widetilde{h^{s_0+tw}}$. Note that as long as the output of $\phi_i(SK)$ satisfies the distribution of $\mathcal{SK}$, it can be taken as a result of the key update, i.e., $\phi_i(SK) = \widetilde{h^{s_0+tw}}$.

(2) Compute $\widetilde{K}'_i = e(g^{\alpha}, \widetilde{SK}_i) = e(g, h)^{\alpha \cdot \widetilde{s_0+tw}}$ as the output of leakage function f. Note that $\widetilde{K}'_i \in \mathbb{G}_T$ and $\mid \widetilde{K}'_i \mid$ should be less than λ. Therefore we should fix the constant l according to the size of group $\mathbb{G}_1$ and $\mathbb{G}_T$. For example, if we set $\parallel \mathbb{G}_1 \parallel = 256$ and $\parallel \mathbb{G}_T \parallel = 3072$, then l will be a constant satisfying $l \geq 16$.

When $\mathcal{A}$ supplies (m, r) pairs to $\mathcal{B}$, $\mathcal{B}$ computes $(C, Z) \leftarrow \text{Enc}(pk, m; r)$ where $C = (g^{\alpha})^r$, $Z = K \cdot m$ and $K = (e(g, h)^{\alpha \cdot s_0})^r$. Then $\mathcal{B}$ computes and returns $\widetilde{M} = Z/(\widetilde{K}'_i)^r$. Note that $\mathcal{B}$ produces the right distribution.

- By receiving $\mathcal{A}$'s challenge plaintexts M_0, M_1, $\mathcal{B}$ forwards them to its own challenger and returns the corresponding challenge ciphertext CT_b to $\mathcal{A}$.
- $\mathcal{B}$ outputs whatever $\mathcal{A}$ does.

□

We can observe that $\mathcal{B}$ simulates perfectly the environment for $\mathcal{A}$, and the advantage of $\mathcal{B}$ to break CML security is the same as $\mathcal{A}$ to break CLT security. Therefore, we can conclude that the scheme is chosen-plaintext secure in the CLT model. Then we can obtain an IND-CCA CLT-secure public-key encryption scheme using the *Naor-Yung double encryption paradigm* [56].

3.3 The efficient construction resisting less tampering queries

In this section, we present our IND-CCA public-key encryption scheme secure in CLT model with less tampering times. At a high level, we achieve chosen-ciphertext security in the CML/CLT model by introducing OT-LF to the public-key encryption scheme in section 6.3.3 of [55]. Since OT-LF and Wichs's scheme are both very efficient, the

resulting scheme is also quite efficient. In fact, this construction can be viewed as a concrete instantiation of a framework for building IND-CCA public-key encryption schemes in the CML model [62]. We provide a different way of proving the security.

The construction Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ is a pairing group of prime order q, where q is $poly(n)$-bits long and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $g, h, e(g,h)$ be generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively. Let LF=($LF.Gen$, $LF.Eval$, $LF.LTag$,$LF.DITag$) be a $(\mathbb{G}_T, l_{LF})$-OT-LF. Let $Ext : \mathbb{G}_T \times \{0,1\}^d \to \{0,1\}^m$ be an average-case ($\log q$-l_{LF}, δ)-strong extractor where δ is negligible in n and $\log q$-l_{LF}-$m \geq \omega(\log(n))$. Let $l \geq 5$ be some constant. Our scheme is parameterized by above parameters as well as the security parameter n.

KeyGen(1^n):	The key generation algorithm samples $\boldsymbol{\alpha}, s_0 \leftarrow \mathbb{Z}_q^l$ and $\boldsymbol{w} \leftarrow span(\boldsymbol{\alpha})^{\perp}$ and runs $LF.Gen(1^n)$ to obtain F_{pk}. Then it sets $PK = (g^{\boldsymbol{\alpha}}, e(g,h)^{\boldsymbol{\alpha} \cdot s_0}, h^{\boldsymbol{w}}, F_{pk})$ and $SK = h^{s_0}$.
Enc(pk, M):	Given a public key $PK = (g^{\boldsymbol{\alpha}}, e(g,h)^{\boldsymbol{\alpha} \cdot s_0}, h^{\boldsymbol{w}}, F_{pk})$, to encrypt a message $M \in \{0,1\}^m$, the encryption algorithm samples $r \leftarrow \mathbb{Z}_q, rand \leftarrow \{0,1\}^d, t_c \leftarrow \mathcal{T}_c$ and computes $C = (g^{\boldsymbol{\alpha}})^r, K = (e(g,h)^{\boldsymbol{\alpha} \cdot s_0})^r, \Psi = Ext(K, rand) \oplus M$, and $\Pi = LF_{F_{pk},t}(K)$ where $t = (t_a, t_c)$ and $t_a = (C, \Psi, rand)$. Then it outputs $CT = (C, \Psi, rand, \Pi, t_c)$ as ciphertext.
Dec(SK, CT):	Given a secret key $SK = h^s$, to decrypt a ciphertext $CT = (C, \Psi, rand, \Pi, t_c)$, the decryption algorithm computes $K' = e(C, SK)$ and checks whether $\Pi = LF_{F_{pk},t}(K')$ where $t = ((C, \Psi, rand), t_c)$. If $\Pi = LF_{F_{pk},t}(K')$, the decryption algorithm outputs $M' = \Psi \oplus Ext(K', rand)$. Otherwise, the decryption algorithm rejects with $\perp$.
KeyUpdate(PK, SK):	Given a secret key $SK = h^s$ and a public key $PK = (g^{\boldsymbol{\alpha}}, e(g,h)^{\boldsymbol{\alpha} \cdot s_0}, h^{\boldsymbol{w}}, F_{pk})$, the update algorithm samples $t \leftarrow \mathbb{Z}_q$ and outputs a new key $SK' = h^s \cdot (h^{\boldsymbol{w}})^t$.

Let SK_i be the decryption key obtained by applying the update algorithm i times. Since $SK_i = h^{s_0 + t\boldsymbol{w}}$ for some $t \in \mathbb{Z}_q$, we have $e((g^{\boldsymbol{\alpha}})^r, SK_i) = (e(g,h)^{\boldsymbol{\alpha} \cdot s_0})^r$. Then the correctness holds.

Security in CML model Firstly, we prove that our scheme is chosen-ciphertext secure in the CML model.

Theorem 3.2 *Under the SXDH assumption, the above encryption scheme is secure against chosen-ciphertext attacks in the CML model with leakage amount* λ=$\log q$-$(m + l_{LF} + \omega(\log n))$.

Proof For any PPT adversary $\mathcal{A}$, we first define a sequence of games each played between a simulator and $\mathcal{A}$. Since $\mathcal{A}$ is PPT, we assume that it makes up to Q_d decryption queries and Q_u update queries where Q_d and Q_u are both polynomial in n. □

Game$_0$: This is the original security game defined in definition 3.1. To be more precisely, the simulator samples $\boldsymbol{\alpha}, s_0 \leftarrow \mathbb{Z}_q^l$, $\boldsymbol{w} \leftarrow span(\boldsymbol{\alpha})^{\perp}$. Then it computes $(F_{pk}, F_{td}) \leftarrow LF.Gen(1^n)$ and sets $PK = (g^{\boldsymbol{\alpha}}, e(g,h)^{\boldsymbol{\alpha} \cdot s_0}, h^{\boldsymbol{w}}, F_{pk})$ and $SK = h^{s_0}$. When $\mathcal{A}$ submits an update query, the simulator computes a new secret key $SK' = SK \cdot h^{t\boldsymbol{w}}$ where t is chosen uniformly at random from $\mathbb{Z}_q$ and sets SK to be SK'. It will also answer the leakage oracle queries and the decryption oracle queries honestly with SK. To generate the challenge ciphertext, the simulator samples $b \leftarrow \{0,1\}$, $t_c^* \leftarrow \mathcal{T}_c$, $r^* \leftarrow \mathbb{Z}_q$, $rand \leftarrow \{0,1\}^d$, computes $C^* = (g^{\boldsymbol{\alpha}})^{r^*}$, $K^* = (e(g,h)^{r^* \boldsymbol{\alpha} \cdot s_0})$, $\Psi^* = Ext(K^*, rand^*) \oplus M_b$, $t^* = ((C^*, \Psi^*, rand^*), t_c^*)$, $\Pi^* = LF_{F_{pk}, t^*}(K^*)$ and sets $CT^* = (C^*, \Psi^*, rand^*, \Pi^*, t_c^*)$. We say the adversary $\mathcal{A}$ wins the game if it can output a bit b' such that $b' = b$.

Game$_1$ This is the same as **Game**$_0$ except following changes. In this game, the simulator samples $\boldsymbol{s}^* \leftarrow \mathbb{Z}_q^l$,and sets $PK = (g^{\boldsymbol{\alpha}}, e(g,h)^{\boldsymbol{\alpha} \cdot s^*}, h^{\boldsymbol{w}}, F_{pk})$. The simulator generates the challenge ciphertext CT^* as **Game**$_0$ except setting $K^* = e(C^*, h^{s^*})$ rather than $K^* = (e(g,h)^{r^* \boldsymbol{\alpha} \cdot s_0})$. In addition, the simulator generates the initial and updated secret keys as $SK = h^{s^* + t\boldsymbol{w}}$ where t is chosen uniformly at random from $\mathbb{Z}_q$ each time.

Game$_1$ and **Game**$_0$ are identical since in **Game**$_1$ we only change s_0 to s^* in generating public/secret key and challenge ciphertext and the correctness holds.

Game$_2$ In this game, the simulator selects a matrix $\mathbf{W}$ that is uniform over $Rk_{l-2}(\mathbb{Z}_q^{l \times (l-2)} \mid col \in span((\alpha))^{\perp})$ and computes each secret keys via $SK = h^{s^* + \mathbf{W} \cdot \boldsymbol{t}}$ for uniform $\boldsymbol{t} \in \mathbb{Z}_q^{l \times (l-2)}$. In addition, $\boldsymbol{w}$ of $h^{\boldsymbol{w}}$ is set to be the first column of $\mathbf{W}$.

Game$_2$ and **Game**$_1$ are computational indistinguishable by the 1-rank-hiding assumption. To see this, assuming $\mathcal{A}$ achieves a non-negligible difference in probability of success between **Game**$_1$ and **Game**$_2$. Then we can construct an efficient distinguisher $\mathcal{B}$ to break the 1-rank-hiding assumption.

We assume that $\mathcal{B}$ is given $g^{\mathbf{X}}$ where $\mathbf{X}$ is chosen uniformly at random from either $Rk_{l-2}(\mathbb{Z}_q^{(l-1) \times (l-2)})$ or $Rk_1(\mathbb{Z}_q^{(l-1) \times (l-2)})$. Then $\mathcal{B}$ simulates the environment for $\mathcal{A}$. It samples $\boldsymbol{\alpha}, \boldsymbol{s}^* \leftarrow \mathbb{Z}_q^l$ and computes $(F_{pk}, F_{td}) \leftarrow LF.Gen(1^n)$. Let $\mathbf{B}$ be a matrix in $\mathbb{Z}_q^{l \times (l-2)}$ whose columns form a basis of $span(\boldsymbol{\alpha})^{\perp}$. Then $\mathcal{B}$ computes $h^{\mathbf{W}} = h^{\mathbf{B} \cdot \mathbf{X}}$ and sets $h^{\boldsymbol{w}}$ to be the first column of $h^{\mathbf{W}}$. Note that, this can be done efficiently since $\mathbf{B}$ is of "exponent". To generate secret keys, $\mathcal{B}$ computes $SK = h^{s^*} \cdot h^{\mathbf{W} \cdot \boldsymbol{t}}$ for $vect \leftarrow \mathbb{Z}_q^{l-2}$ each time. It can generate the challenge ciphertext and the public key with above parameters and it can also answer decryption queries and leakage queries with its secret keys. It is easy to see that when $\mathbf{X}$ is uniform over $Rk_1(\mathbb{Z}_q^{(l-1) \times (l-2)})$, all columns of $\mathbf{W}$ are in $span(\boldsymbol{w})$ and thus $\mathcal{B}$ properly simulates **Game**$_1$. Otherwise, $\mathcal{B}$ properly simulates **Game**$_2$ since $\mathbf{W}$ is uniform over $Rk_{l-2}(\mathbb{Z}_q^{l \times (l-2)} \mid col \in span(\boldsymbol{\alpha})^{\perp})$ now. Therefore, the non-negligible advantage of $\mathcal{A}$ implies the non-negligible advantage of $\mathcal{B}$.

Game$_3$ In this game, the simulator generates a dishonest challenge ciphertext instead. In the beginning, the simulator chooses $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leftarrow Rk_2(\mathbb{Z}_q^{l \times 2})$ and $\mathbf{W} \leftarrow Rk_{l-2}(\mathbb{Z}_q^{l \times (l-2)} \mid col \in span(\boldsymbol{\alpha}, \boldsymbol{\beta})^{\perp})$. Then it computes the public key and secret keys using above parameters. To generate the challenge ciphertext, the simulator computes $C^* = g^{\boldsymbol{\beta}}$ and continues computing the remaining parts using C^* as before.

Game$_3$ and **Game**$_2$ are also computational indistinguishable by the 1-rank-hiding assumption. We prove this by constructing an efficient distinguisher $\mathcal{B}$ that breaks the 1-rank-hiding assumption based on $\mathcal{A}$'s non-negligible difference in probability of success between **Game**$_2$ and **Game**$_3$. Note that, we can view C^* in **Game**$_2$ as $C^* = g^{\boldsymbol{\beta}}$ where $\boldsymbol{\beta} \leftarrow span(\boldsymbol{\alpha})$.

$\mathcal{B}$ is given $g^{\mathbf{X}}$ where $\mathbf{X}$ is chosen uniformly at random from either $Rk_2(\mathbb{Z}_q^{2\times 2})$ or $Rk_1(\mathbb{Z}_q^{2\times 2})$. Then $\mathcal{B}$ simulates the environment for $\mathcal{A}$. It samples $\boldsymbol{s}^* \leftarrow \mathbb{Z}_q^l$, $\mathbf{W} \leftarrow Rk_{l-2}(\mathbb{Z}_q^{l\times(l-2)})$ and computes $(F_{pk}, F_{td}) \leftarrow LF.Gen(1^n)$. Let $\mathbf{B}$ be a matrix in $\mathbb{Z}_q^{l\times 2}$ whose columns form a basis of $span(\mathbf{W})^{\perp}$. Then $\mathcal{B}$ computes $h^{(\alpha,\beta)} = h^{\mathbf{B}\cdot\mathbf{X}}$. Then by Lemma 5, using the parameters and setting $C^* = g^{\boldsymbol{\beta}}$ in generating the challenge ciphertext, $\mathcal{B}$ can simulates **Game**$_3$ if $\mathbb{X}$ is uniform over $Rk_2(\mathbb{Z}_q^{2\times 2})$ and **Game**$_2$ otherwise.

Game$_4$ This is the same as **Game**$_3$ except that $\mathbf{W}$ is chosen from $U(\mathbb{Z}_q^{l\times(l-2)} \mid col \in span(\boldsymbol{\alpha}, \boldsymbol{\beta})^{\perp})$. By Lemma 3, it is easy to see that **Game**$_4$ and **Game**$_3$ are indistinguishable.

Game$_5$ In this game, we modify the way of generating the challenge ciphertext. In more detail, the simulator keeps the trapdoor F_{td} generated together with F_{pk} and computes t_c^* with $LF.LTag(F_{td}, t_a^*$ where $t_a^* = (C^*, \Psi^*, rand^*)$. The computational indistinguishability of **Game**$_4$ and **Game**$_5$ follows directly from the indistinguishability of LF by a straightforward reduction.

Game$_6$ This is the same as **Game**$_5$ except that the simulator will directly answer $\perp$ if $\mathcal{A}$ queries the decryption oracle with a ciphertext $CT = (C, \Psi, rand, \Pi, t_c)$ such that t=$((C, \Psi, rand), t_c)$=$((C^*, \Psi^*, rand^*), t_c^*)$=$t^*$. We call such tag a *copied tag* for convenient. One can see that with all but negligible probability the simulator in **Game**$_5$ will also reject a copied tag that $\mathcal{A}$ can query and thus $\mathcal{A}$ cannot distinguish between **Game**$_5$ and **Game**$_6$ with non-negligible advantage. We show this by considering the following cases:

1. A decryption query with a copied tag is submitted before the challenge ciphertext is generated. This can occur with only negligible probability since $\mathcal{A}$ cannot even predicate $rand^*$ with non-negligible probability.
2. A decryption query with a copied tag is submitted after the challenge ciphertext is generated and $\Pi = \Pi^*$. This implies $CT = CT^*$ and should be rejected since $\mathcal{A}$ is not allowed to ask for the decryption of the challenge ciphertext.
3. A decryption query with a copied tag is submitted after the challenge ciphertext is generated and $\Pi \neq \Pi^*$. Since $C = C^* = g^{\boldsymbol{\beta}}$ and $(\boldsymbol{s}' - \boldsymbol{s}^*) \cdot \boldsymbol{\beta} = 0$ where $\boldsymbol{s}'$ is the exponent of the secret key used when answering this decryption query, it follows that $K = K^*$. and thus $LF_{F_{pk},t}(K) = LF_{F_{pk},t^*}(K^*) = \Pi^*$. So this query would be rejected by the decryption algorithm.

Game$_7$ This is the same as **Game**$_6$ except that the simulator further rejects if $\mathcal{A}$ queries the decryption oracle with a ciphertext that has a non-injective tag. $\mathcal{A}$ cannot distinguish between **Game**$_6$ and **Game**$_7$ with non-negligible advantage since the simulator in **Game**$_6$ will also reject a ciphertext with non-injective tag that $\mathcal{A}$ can query with all but negligible probability. We show this by considering the following cases:

1. A decryption query with a non-injective copied tag is submitted. The simulator in $\mathbf{Game}_6$ also rejects this tag since it will reject all copied tags.
2. A decryption query with a non-injective non-copied tag is submitted. This cannot occur with non-negligible probability due to the evasiveness of LF. We show this by constructing an algorithm $\mathcal{B}$ that can break the evasiveness of LF if $\mathcal{A}$ submits a ciphertext that has a non-injective non-copied tag with a non-negligible probability ϵ. $\mathcal{B}$ is given F_{pk} in the beginning and then it can simulate the entire environment of $\mathbf{Game}_6$ for $\mathcal{A}$. To generate the challenge ciphertext, $\mathcal{B}$ first computes $(C^*, \Psi^*, rand^*)$ and queries its lossy tag generation oracle once with $t_a^* = (C^*, \Psi^*, rand^*)$ to obtain t_c^*. Finally, $\mathcal{B}$ chooses $i \leftarrow [Q_d]$ and outputs the tag $t = ((C, \Psi, rand), t_c)$ extracted from $\mathcal{A}$'s i-th decryption query $CT = (C, \Psi, rand, \Pi, t_c)$. Clearly, if a ciphertext with non-injective non-copied tag is submitted with non-negligible probability ϵ, $\mathcal{B}$ can generate a new non-injective tag with a probability of at leas $\frac{\epsilon}{Q_d}$.

$\mathbf{Game}_8$ This is the same as $\mathbf{Game}_7$ except the following two modifications. In this game, the simulator rejects a decryption oracle query $CT = (C, \Psi, rand, \Pi, t_c)$ directly if the exponent of C is not in $span(\boldsymbol{\alpha})$. It also generates secret keys by sampling a new vector $\boldsymbol{s}$ uniformly from the affine subspace $\{\boldsymbol{x} \mid \boldsymbol{x} \cdot \boldsymbol{\alpha} = \boldsymbol{s}^* \cdot \boldsymbol{\alpha}\}$ and set $SK = g^{\boldsymbol{s}}$.

To see the probabilities that $\mathcal{A}$ wins in $\mathbf{Game}_7$ and $\mathbf{Game}_8$ have only a negligible difference, we define the following games Game_{H_i} for $i \in [0, Q_u + 1]$. For convenient, we define a round to be a time period between two consecutive update queries. Also, the first round is the time period before the first update query and the last round is the time period after the last update query.

Game_{H_i} Game_{H_i} behaves like $\mathbf{Game}_8$ in the first i rounds and like $\mathbf{Game}_7$ in the remaining rounds. More precisely, Game_{H_i} is the same as $\mathbf{Game}_7$, except that the above mentioned two modifications are applied to the first i rounds.

It is easy to see that Game_{H_0} is identical to $\mathbf{Game}_7$ and $\text{Game}_{H_{Q_u+1}}$ is identical to Game $\mathbf{Game}_8$. And since Q_u is polynomial in n, it is suffice to prove the following claim. We defer its proof until later.

Claim 3.5 Denote S_i, S_{i+1} to be events that $\mathcal{A}$ wins Game_{H_i} and $\text{Game}_{H_{i+1}}$, respectively. Then we have $|\Pr[S_i] - \Pr[S_{i+1}]| \neq negl(n)$ for any $i \in [0, Q_u]$.

$\mathbf{Game}_9$: This is the same as $\mathbf{Game}_8$ except that M_b is masked by a uniform string over $\{0, 1\}^m$ in the challenge ciphertext.

To see the indistinguishability of $\mathbf{Game}_8$ and $\mathbf{Game}_9$, we first lower bound the average-case min-entropy of K^*. Since all secret keys are chosen uniformly at random from the affine subspace $\{\boldsymbol{x} \mid \boldsymbol{x} \cdot \boldsymbol{\alpha} = \boldsymbol{s}^* \cdot \boldsymbol{\alpha}\}$, they reveal nothing about $\boldsymbol{s}^*$ except the value of $\boldsymbol{s}^* \cdot \boldsymbol{\alpha}$ which has already been determined by PK. Thus, all leakage oracle queries and decryption oracle queries provide no additional information about $\boldsymbol{s}^*$. Therefore, we have:

$$\begin{aligned} \tilde{H}_\infty(K^* \mid View(\mathcal{A})) &= \tilde{H}_\infty(K^* \mid PK, CT^*) \\ &= \tilde{H}_\infty(K^* \mid \boldsymbol{s}^* \cdot \boldsymbol{\alpha}, \Pi^*) \\ &\geq \tilde{H}_\infty(K^* \mid \boldsymbol{s}^* \cdot \boldsymbol{\alpha}) - l_{LF} \end{aligned} \tag{1}$$

$$= \tilde{H}_\infty(\boldsymbol{\beta}^* \cdot \boldsymbol{s}^* \mid \boldsymbol{s}^* \cdot \boldsymbol{\alpha}) - l_{LF}$$
$$= \log(q) - l_{LF} \quad (2)$$

Here $View(\mathcal{A})$ denotes the view of $\mathcal{A}$ during the entire game. The inequality in (1) follows from the lossiness of LF and Lemma 6, and the equality in (2) follows from Lemma 2 as $\boldsymbol{\beta} \notin span(\boldsymbol{\alpha})$. Because Ext is an average-case ($\log q$-l_{LF}, δ)-strong extractor, we have that $Ext(K^*, rand^*)$ is statistically indistinguishable from U_m and the indistinguishability of $\mathbf{Game}_8$ and $\mathbf{Game}_9$ follows.

We conclude the proof by observing that in $\mathbf{Game}_9$, the adversary's advantage to distinguish the challenge ciphertext is 0, since in $\mathbf{Game}_9$ M_b is masked by a uniform string that is independent of the view of $\mathcal{A}$.

Now, it remains to prove Claim 3.5:

Proof We further define the following game for $i \in [0, Q_u]$.
$\text{Game}_{H_i'}$: This is the same as Game_{H_i} except that the simulator further rejects a queried ciphertext $CT = (C, \Psi, rand, \Pi, t_c)$ directly if the exponent of C is not in $span(\boldsymbol{\alpha})$ in the $(i+1)$-th round.

We call a ciphertext that will be rejected in $\text{Game}_{H_i'}$ but not in Game_{H_i} an invalid ciphertext. More precisely, an invalid ciphertext $CT = (C, \Psi, rand, \Pi, t_c)$ satisfies $\Pi = LF_{F_{pk},t}(e(C, SK_{i+1}))$ and $\boldsymbol{\gamma} \notin span(\boldsymbol{\alpha})$ where $t = ((C, \Psi, rand), t_c)$ is an injective tag, SK_{i+1} is the secret key used in the $(i+1)$-th round and $\boldsymbol{\gamma}$ is the exponent of C. Let F be the event that $\mathcal{A}$ submits an invalid ciphertext when querying the decryption oracle during the $(i+1)$-th round. It is obvious that $\text{Game}_{H_i'}$ proceeds identically to Game_{H_i} until F occurs. As stated in [63], we have $|\Pr[S_i'] - \Pr[S_i]| \leq \Pr[F]$ where S_i' is the event that $\mathcal{A}$ wins in $\text{Game}_{H_i'}$ and S_i is the event that $\mathcal{A}$ wins in Game_{H_i}. Therefore, indistinguishability of $\text{Game}_{H_i'}$ and Game_{H_i} holds as long as $\Pr[F]$ is negligible. □

Claim 3.6 $\Pr[F]$ *is negligible in* n.

Proof To prove Claim 3.6, let us condition on fixed public parameters as well as fixed values of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\mathbf{W}$, F_{pk}, $\boldsymbol{s}^* \cdot \boldsymbol{\alpha}$ and $\mathcal{A}$'s coins. We also fix all secret keys used in the first i rounds. These values completely determine the public key, and all the priori queries of $\mathcal{A}$ as well as the corresponding responses of the simulator in the first i rounds. Also, whether $\mathcal{A}$ makes the challenge query in the first i rounds, and if so, the corresponding values of M_0 and M_1 are determined. If $\mathcal{A}$ makes the challenge query in the first i rounds, we further fix the challenge ciphertext CT^*. Anyway, the view of $\mathcal{A}$ in the first i rounds is determined, and whether it enters the $(i+1)$-th round is also determined. Assume that $\mathcal{A}$ enters the $(i+1)$-th round; otherwise $\Pr[F] = 0$ in the conditional probability space which is what we need. Then we further fix the value of $\boldsymbol{t}_{i+1}$ which is employed to compute the secret key used in the $(i+1)$-th round by setting $Sk_{i+1} = g^{\boldsymbol{s}^* + \mathbf{W} \cdot \boldsymbol{t}_{i+1}}$. During the $(i+1)$-th round, $\mathcal{A}$ is only able to make the following queries:

- **Challenge queries** We fix the challenge ciphertext CT^* and send it to $\mathcal{A}$.
- **Leakage queries**. We fix the value of the response to $\mathcal{A}$'s query and send it to $\mathcal{A}$.
- *Decryption Queries*. Suppose $\mathcal{A}$ submit a ciphertext $CT = (C, \Psi, rand, \Pi, t_c)$ and let $C = g^{\boldsymbol{\gamma}}$. If $\boldsymbol{\gamma} \in span(\boldsymbol{\alpha})$,response to this query is determined by PK and otherwise it will be exactly $\perp$. □

Therefore, all $\mathcal{A}$'s queries as well as responses to them in the $(i+1)$-th round are determined or fixed setup by setup. Specially, all $\mathcal{A}$'s decryption queries are determined in the beginning conditioned on the values we have fixed. Assume $CT = (C, \Psi, rand, \Pi, t_c)$ among them is an invalid ciphertext. We write $C = g^{\boldsymbol{\gamma}}$ and have $\boldsymbol{\gamma} \notin span(\boldsymbol{\alpha})$, $t = ((C, \Psi, rand), t_c)$ is an injective tag, and $\Pi = LF_{F_{pk},t}(e(g,h)^{\boldsymbol{\gamma}\cdot(\boldsymbol{s}^*+\boldsymbol{v}_{i+1})})$. Since $LF_{F_{pk}}$ is an injective function, $\mathcal{A}$ must give a tuple $(\boldsymbol{\gamma}, \boldsymbol{\gamma}\cdot\boldsymbol{s}^*)$ to generate CT.

For any $\nu \in \omega(\log(n))$,assume $\boldsymbol{s}^*$ has min-entropy $(l-2)\log(q)+\nu$ in the above conditional probability space Ω. Then for any $\boldsymbol{\gamma} \in \mathbb{Z}_q^l \backslash span(\boldsymbol{\alpha})$ and $z \in \mathbb{Z}_q$, we have $\Pr[\boldsymbol{\gamma}\cdot\boldsymbol{s}^* = z]$ is negligible. To see this, we assume there exists $\boldsymbol{\gamma} \in \mathbb{Z}_q^l \backslash span(\boldsymbol{\alpha})$ such that $\Pr[\boldsymbol{\gamma}\cdot\boldsymbol{s}^* = z] = \frac{1}{p}$ for some polynomial p and denote by y the fixed value of $\boldsymbol{\alpha}\cdot\boldsymbol{s}^*$. Since $\boldsymbol{\gamma} \notin span(\boldsymbol{\alpha})$, $\boldsymbol{s}^*$ can be predicated with probability $\frac{1}{q^{l-2}}$ conditioned on the equations:

$$\begin{cases} \boldsymbol{\alpha}\cdot\boldsymbol{s}^* = y \\ \boldsymbol{\gamma}\cdot\boldsymbol{s}^* = z \end{cases}$$

Therefore,$\boldsymbol{s}^*$ can be predicated with probability of at least $\frac{1}{p\cdot q^{l-2}}$ in Ω which means $H_\infty(\boldsymbol{s}^*) \neq (l-2)\log(q) + \log(p) < (l-2)\log(q)+\nu$ and contradicts the assumption that $\boldsymbol{s}^*$ has min-entropy $(l-2)\log(q)+\nu$. Therefore, $\mathcal{A}$ can only generate an invalid ciphertext with negligible probability in Ω. And since $\mathcal{A}$ can only submit at most Q_d ciphertexts in the $(i+1)$-th round for a polynomial Q_d, F can occur in Ω with only negligible probability.

It remains to lower bound the min-entropy of $\boldsymbol{s}^*$. For convenient, we write $\lambda = \log(q) - (m + l_{LF} + \mu)$ for some $\mu \in \omega(\log(n))$ and denote by Λ_{i+1} all leakage information leaked during the $(i+1)$-th round which has a length of at most λ. Since we are concerning whether $\mathcal{A}$ can submit an invalid ciphertext during the $(i+1)$-th round, we only need to consider the entropy of $\boldsymbol{s}^*$ conditioned on the view of $\mathcal{A}$ before the $(i+1)$-th update query. Further, the view of $\mathcal{A}$ in the first i rounds except the challenge ciphertext, the response to $\mathcal{A}$'s decryption oracle queries in the $(i+1)$-th round are all determined by PK and thus provide no additional information. Therefore, we have

$$H_\infty(\boldsymbol{s}^* \mid View(\mathcal{A})) = H_\infty(\boldsymbol{s}^* \mid PK, CT^*, \Lambda_{i+1}) = H_\infty(\boldsymbol{s}^* \mid \boldsymbol{\alpha}\cdot\boldsymbol{s}^*, \Psi^*, \Pi^*, \Lambda_{i+1}).$$

By Lemma 7, we have:

$\Pr[H_\infty(\boldsymbol{s}^* \mid \boldsymbol{\alpha}\cdot\boldsymbol{s}^*, \Psi^*, \Pi^*, \Lambda_{i+1}) \leq H_\infty(\boldsymbol{s}^* \mid \boldsymbol{\alpha}\cdot\boldsymbol{s}^*) - (\log(q)-\mu) - \frac{\mu}{2}] \leq 2^{-\frac{\mu}{2}}$

Since $H_\infty(\boldsymbol{s}^* \mid \boldsymbol{\alpha}\cdot\boldsymbol{s}^*) = (l-1)\log(q)$, we have:

$\Pr[H_\infty(\boldsymbol{s}^* \mid View(\mathcal{A})) \leq (l-2)\log(q) + \frac{\mu}{2}] \leq 2^{-\frac{\mu}{2}}$

which implies that the above assumption saying $\boldsymbol{s}^*$ has a large min-entropy in Ω hold with all but negligible probability and that completes the proof of Claim 3.6.

$\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$ are identical except that the secret key in the $(i+1)$-th round is sampled from different distributions. Thus, we show the indistinguishability of $\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$ by constructing the following computationally unbounded

simulator $\mathcal{B}$ that can simulate the environment of either $\text{Game}_{H_i'}$ or $\text{Game}_{H_{i+1}}$ when given statistically indistinguishable distributions. For convenient, we fix $\mathcal{A}$'s coins and prove that $\mathcal{A}$ can not distinguish between $\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$ even when equipped with any randomness.

In the beginning, $\mathcal{B}$ samples $\boldsymbol{\alpha}, \boldsymbol{s}^* \leftarrow \mathbb{Z}_q^l$, $\boldsymbol{w} \leftarrow span(\boldsymbol{\alpha})^{\perp}$ and computes $(F_{pk}, F_{td}) \leftarrow LF.Gen(1^n)$. Then it can generate the public key and secret keys in the first i rounds properly as in $\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$. Suppose $\mathcal{A}$ does not make the challenge query in the first i rounds; otherwise, since the responses to all decryption oracle queries in the $(i+1)$-th round are determined by PK and $\mathcal{A}$ is not allowed to query the leakage oracle after the challenge phase, $\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$ behaves identically and thus $\mathcal{A}$ cannot distinguish them with any advantage. Then $\mathcal{B}$ can simulate the environment properly in the first i rounds. Now suppose $\mathcal{A}$ enters the $(i+1)$-th round; if not, $\mathcal{A}$'s view is also identical in both games. $\mathcal{B}$ can answer $\mathcal{A}$'s decryption oracle queries in the $(i+1)$-th round with the public key since responses to them are all determined by PK. To handle $\mathcal{A}$'s leakage oracle queries, $\mathcal{B}$ first encode all leakage $\mathcal{A}$ asks in the $(i+1)$-th round as a single function F. $\mathcal{B}$ can do this since it is computationally unbounded and is able to acquire $\mathcal{A}$'s strategy by examining all possible cases. Let $\mathcal{B}$ be a matrix in $\mathbb{Z}_q^{l\times(l-1)}$ whose columns form a basis of $span(\boldsymbol{\alpha})^{\perp}$. Then $\mathcal{B}$ queries $f : \mathbb{Z}_q^{l-1} \to \{0,1\}^{\lambda}$ where $f(\boldsymbol{x}) = F(g^{\boldsymbol{s}^*+r\boldsymbol{w}+\mathbf{B}\cdot\boldsymbol{x}})$ for $r \leftarrow \mathbb{Z}_q$, and receives a sample $(f(\boldsymbol{x}), \mathbf{A})$ where $\mathbf{A} \leftarrow \mathbb{Z}_q^{(l-1)\times(l-3)}$ and $\boldsymbol{x}$ is chosen uniformly from either $\mathbb{Z}_q^{l-1}$ or $colspan(\mathbf{A})$. Now $\mathcal{B}$ can answer $\mathcal{A}$'s leakage oracle queries with $f(\boldsymbol{x})$. It also sets $\mathbf{W} = [\boldsymbol{w} \parallel \mathbf{B}\cdot\mathbf{A}]$ and samples $\boldsymbol{\beta} \leftarrow colspan(\mathbf{W})^{\perp}$. By Corollary 2 and Lemma 3, $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{W})$ generated above is statistically close to that of $\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$. Therefore, $\mathcal{B}$ can simulate the remaining rounds properly with $\mathbf{W}$ and $\boldsymbol{\beta}$.

When $\boldsymbol{x}$ is chosen uniformly from $\mathbb{Z}_q^{l-1}$, $\mathbf{B}\cdot\boldsymbol{x}$ is a uniform vector in $span(\boldsymbol{\alpha})^{\perp}$ and thus the secret key used in the $(i+1)$-th round is distributed identical to that of $\text{Game}_{H_{i+1}}$, which means $\mathcal{B}$ properly simulates $\text{Game}_{H_{i+1}}$ in this case. In contrast, when $\boldsymbol{x} \leftarrow colspan(\mathbf{A})$, $r\boldsymbol{w} + \mathbf{B}\cdot\boldsymbol{x}$ is a uniform vector in $colspan(\mathbf{W})$ and thus $\mathcal{B}$ properly simulates $\text{Game}_{H_i'}$ in this case. Since $l \geq 5$, we have $\lambda = \log(q) - l_{LF} - m - \omega(\log(n)) \leq (l-4)\log(q) - \omega(\log(n))$. By Lemma 1, the indistinguishability of $\text{Game}_{H_i'}$ and $\text{Game}_{H_{i+1}}$ follows. Therefore, Game_{H_i} and $\text{Game}_{H_{i+1}}$ are indistinguishable to $\mathcal{A}$ since they are both indistinguishable from $\text{Game}_{H_i'}$. This completes the proof of Theorem 3.2.

Security in CLT model In the following, we prove that our scheme is IND-CCA secure in the CLT model. We can simulate tampering queries by only leaking a bounded amount of information from the secret key as a decryption result for a given tampered key. Hence, security in CLT model follows from the security in CML model.

Theorem 3.3 The above encryption scheme is (λ', t')-secure against chosen-cipher text attacks in the CLT model, where

$$\lambda' \leq \lambda - t'm \leq log(q) - l_{LF} - (1+t')m - \omega(log(n)) \quad and \quad t' \leq \lfloor (log(q) - l_{LF})/m \rfloor - 1$$

Proof To prove this, assuming an adversary $\mathcal{A}$ can break IND-CCA CLT security of the scheme, then we construct an adversary $\mathcal{B}$ to break IND-CCA CML security of

the scheme. Adversary $\mathcal{B}$ simulates environment for $\mathcal{A}$ and uses $\mathcal{A}$ as a black box as follows:

- $\mathcal{B}$ is given (PK, SK) from its challenger and forwards them to $\mathcal{A}$.
- *Leakage Queries*. Whenever $\mathcal{A}$ asks for a leakage query, $\mathcal{B}$ submits this query to its challenger and returns the answer directly to $\mathcal{A}$.
- *Tampering Queries*. Upon receiving a tampering query $\mathcal{O}(tamper, \phi_i, CT = (C, \Psi, rand, \Pi, t_c))$ where $\phi_i \in \Phi_{sk}$ is a tampering function and c is a ciphertext chosen by $\mathcal{A}$, $\mathcal{B}$ firstly checks whether $T + 1 \leq t'$ and returns the result by submitting a leakage function f to its own leakage oracle such that:

  (1) Set $\widetilde{SK}_i = \phi_i(SK) = \widetilde{h^{s+tw}}$.
  (2) Compute $\widetilde{K}'_i = e(C, \widetilde{SK}_i) = (e(g, h)^{\alpha \cdot \widetilde{s_0 + tw}})^r$.
  (3) Return $\widetilde{M} = \Psi \oplus Ext(\widetilde{K}'_i, rand)$ as the output of leakage function f. Note that $\mathcal{B}$ produces the right distribution and $| \widetilde{M} | < \lambda$ which gives $\lambda' \leq \lambda - t'm$.

- By receiving $\mathcal{A}$'s challenge plaintexts M_0, M_1, $\mathcal{B}$ forwards them to its own challenger and returns the corresponding challenge ciphertext CT_b to $\mathcal{A}$.
- $\mathcal{B}$ outputs whatever $\mathcal{A}$ does.

We can observe that $\mathcal{B}$ simulates perfectly the environment for $\mathcal{A}$ and the advantage of $\mathcal{B}$ to break CML security is the same as $\mathcal{A}$ to break CLT security. Therefore, we can conclude that the scheme is IND-CCA secure in the CLT model. □

4 Conclusion

In this paper, we focus on how to construct an efficient IND-CCA secure public-key encryption scheme with continuous leakage and tampering resilience. We present a somewhat inefficient IND-CCA secure public-key encryption scheme in CLT model with linear tampering times and show how to construct an efficient IND-CCA secure public-key encryption scheme in CML/CLT model by introducing one-time lossy filter into an efficient IND-CPA secure public-key encryption scheme in the CML model. Unlike Damgård et al.'s scheme, our schemes do not need a securely kept update key to refresh the decryption key. Our second scheme is as efficient as the original scheme of Wichs since one-time lossy filter is much more efficient compared to Wichs's public-key encryption scheme. However, since one-time lossy filter can only tolerate a leakage of size of the encapsulated key which is smaller compared to the secret key of Wichs's scheme, our second scheme has a lower leakage rate than the original scheme. Also, our second scheme can tolerate less tampering queries. We leave it an interesting problem how to construct an efficient continuous leakage and tampering resilient IND-CCA secure public-key encryption schemes with higher leakage rate and more tampering times.

Acknowledgements This project is supported by National Natural Science Foundation of China (No. 61602275), the Open Project of Co-Innovation Center for Information Supply & Assurance Technology, Anhui University (No. ADXXBZ201702), and Shandong Province Higher Educational Science and Technology Program (No. J15LN01).

References

1. Fortis T, Munteanu V, Negru V (2015) A taxonomic view of cloud computing services. Int J Comput Sci Eng 11(1):17–28
2. Gao C, Cheng Q, Li X, Xia S (2018) Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network. Clust Comput. https://doi.org/10.1007/s10,586-017-1649-y
3. Shen J, Gui Z, Ji S, Shen J, Tan H (2018) Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks. J Netw Comput Appl. https://doi.org/10.1016/j.jnca.2018.01.003
4. Gai K, Liu M, Hassan H (2017) Secure cyber incident analytics framework using monte carlo simulations for financial cybersecurity insurance in cloud computing. Concurr Comput Pract Exp 29(7):e3856
5. Bertino E, Paci F, Ferrini R, Shang N (2009) Privacy-preserving digital identity management for cloud computing. IEEE Data Eng Bull 32:21–27
6. Xu J, Wei L, Zhang Y, Wang A, Zhou F, Cz Gao (2018) Dynamic fully homomorphic encryption-based merkle tree for lightweight streaming authenticated data structures. J Netw Comput Appl 107:113–124
7. Joshi J, Bhatti R, Bertino E, Ghafoor A (2004) Access control language for multidomain environments. IEEE Internet Comput 8(6):40–50
8. Zhong H, Zhu W, Xu Y, Cui J (2018) Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage. Soft Comput 22:243–251
9. Hesamifard E, Takabi H, Ghasemi M, Jones C (2017) Privacy-preserving machine learning in cloud. In: CCSW 2017, pp 39–43
10. Li P, Li J, Huang Z, Gao C, Chen W, Chen K (2017) Privacy-preserving outsourced classification in cloud computing. Clust Comput. https://doi.org/10.1007/s10,586-017-0849-9
11. Ding W, Yan Z, Deng R (2017) Secure encrypted data deduplication with ownership proof and user revocation. In: ICA3PP 2017, pp 297–312
12. Li J, Li Y, Chen X, Lee P, Lou W (2015) A hybrid cloud approach for secure authorized deduplication. IEEE Trans Parallel Distrib Syst 26(5):1206–1216
13. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Eurocrypt 2004, pp 506–522
14. Cui J, Zhou H, Zhong H, Xu Y (2018) Akser: attribute-based keyword search with efficient revocation in cloud computing. Inf Sci 423:343–352
15. Lai J, Zhou X, Deng RH, Li Y, Chen K (2013) Expressive search on encrypted data. In: AisaCCS 2013, pp 243–252
16. Xu Y, Wang M, Zhong H, Cui J, Liu L, Franqueira V (2017) Verifiable public key encryption scheme with equality test in 5g networks. IEEE Access 5:12,702–12,713
17. Yang L, Han Z, Huang Z, Ma J (2018) A remotely keyed file encryption scheme under mobile cloud computing. J Netw Comput Appl 106:90–99
18. Zhong H, Cui J, Shi R, Xia C (2016) Many-to-one homomorphic encryption scheme. Secur Commun Netw 9(10):1007–1015
19. Li J, Huang X, Li J, Chen X, Xiang Y (2014) Securely outsourcing attribute-based encryption with checkability. IEEE Trans Parallel Distrib Syst 25(8):2201–2210
20. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D (2007) Provable data possession at untrusted stores. In: CCS 2007. ACM, pp 598–609
21. Li J, Liu Z, Chen X, Xhafa F, Tan X, Wong DS (2015) L-encdb: a lightweight framework for privacy-preserving data queries in cloud computing. Knowl Based Syst 79:18–26
22. Li B, Huang Y, Liu Z, Li J, Tian Z, Yiu SM (2018) Hybridoram: practical oblivious cloud storage with constant bandwidth. Inf Sci. https://doi.org/10.1016/j.ins.2018.02.019
23. Hohenberger S, Rothblum G, shelat A, Vaikuntanathan V (2011) Securely obfuscating re-encryption. Proceedings of the Theory of Cryptography Conference. J Cryptol 24(4):694–719
24. Li J, Li J, Chen X, Jia C, Lou W (2015) Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans Comput 64(2):425–437
25. Liu Q, Guo Y, Wu J, Wang G (2017) Effective query grouping strategy in clouds. J Comput Sci Technol 32(6):1231–1249
26. Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: Advances in Cryptology-CRYPTO 1997. Springer, pp 513–525
27. Gandolff K, Mourtel C, Olivier F (2001) Electromagnetic analysis: concrete results. In: CHES 2001, pp 251–261

28. Biham E, Carmeli Y, Shamir A (2008) Bug attacks. In: Advances in Cryptology-CRYPTO 2008. Springer, pp 221–240
29. Halderman J, Schoen S, Nadia H, Clarkson W, Paul W, Calandrino J, Feldman A, Appelbaum J, Felten E (2008) Lest we remember: cold-boot attacks on encryption keys. In: USENIX Security Symposium 2008, pp 45–60
30. Ristenpart T, Tromer E, Shacham H, Savage S (2009) Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: CCS, pp 199–212 (2009)
31. Zhang Y, Juels A, Reiter M, Ristenpart T (2012) Cross-VM side channels and their use to extract private keys. In: CCS, pp 305–316 (2012)
32. Inci M, Gulmezoglu B, Irazoqui G, Eisenbarth T, Sunar B (2015) Seriously get off my cloud Cross-VM RSA Key Recovery in a Public Cloud. Cryptology ePrint Archive 2015:898
33. Bellare M, Cash D, Miller R (2011) Cryptography secure against related-key attacks and tampering. In: Advances in Cryptology-ASIACRYPT 2011. Springer, pp 486–503
34. Gennaro R, Lysyanskaya A, Malkin T, Micali S, Rabin T (2004) Algorithmic tamper-proof (atp) security: theoretical foundations for security against hardware tampering. In: TCC 2004. Springer, pp 258–277
35. Wee H (2012) Public key encryption against related key attacks. In: PKC 2012. Springer, pp 262–279
36. Akkar ML, Giraud C (2001) An implementation of des and aes, secure against some attacks. In: CHES 2001, pp 309–318
37. Trichina E, De Seta D, Germani L (2002) Simplified adaptive multiplicative masking for aes. In: CHES 2002, pp 187–197
38. Dziembowski S, Pietrzak K (2008) Leakage-resilient cryptography. In: FOCS 2008, pp 293–302
39. Juma A, Vahlis Y (2010) Protecting cryptographic keys against continual leakage. In: Advances in Cryptology-CRYPTO 2010. Springer, pp 41–58
40. Chow S, Dodis Y, Rouselakis Y, Waters B (2010) Practical leakage-resilient identity-based encryption from simple assumptions. In: CCS 2010, pp 152–161
41. Boyle E, Segev G, Wichs D (2011) Fully leakage-resilient signatures. In: Advances in Cryptology-EUROCRYPT 2011. Springer, pp 89–108
42. Halevi S, Lin H (2011) After-the-fact leakage in public-key encryption. In: TCC 2011, pp 474–495
43. Zhang M, Yang B, Takagi T (2013) Bounded leakage-resilient functional encryption with hidden vector predicate. Comput J 56(4):464–477
44. Huang Z, Liu S, Mao X, Chen K, Li J (2017) Insight of the protection for data security under selective opening attacks. Inf Sci 412:223–241
45. Brakerski Z, Kalai Y, Katz J, Vaikuntanathan V (2010) Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage. In: FOCS 2010, pp 501–510
46. Dodis Y, Haralambiev K, Lopez-Alt A, Wichs D (2010) Cryptography against continuous memory attacks. In: FOCS 2010, pp 511–520
47. Shen J, Wang C, Li T, Chen X, Huang X, Zhan ZH (2018) Secure data uploading scheme for a smart home system. Inf Sci. https://doi.org/10.1016/j.ins.2018.04.048
48. Chen X, Li J, Weng J, Ma J, Lou W (2016) Verifiable computation over large database with incremental updates. IEEE Trans Comput 65(10):3184–3195
49. Lewko A, Rouselakis Y, Waters B (2011) Achieving leakage resilience through dual system encryption. In: TCC 2011, pp 70–88
50. Lewko A, Lewko M, Waters B (2011) How to leak on key updates. In: STOC 2011, pp 725–734
51. Dodis Y, Lewko A, Waters B, Wichs D (2011) Storing secrets on continually leaky devices. In: FOCS 2011, pp 688–697
52. Kalai Y, Kanukurthi B, Sahai A (2011) Cryptography with tamperable and leaky memory. In: Advances in Cryptology-CRYPTO 2011. Springer, pp 373–390
53. Damgård I, Faust S, Mukherjee P, Venturi D (2013) Bounded tamper resilience: How to go beyond the algebraic barrier. In: Advances in Cryptology-ASIACRYPT 2013. Springer, pp 140–160
54. Li J, Chen X, Li M, Li J, Lee PP, Lou W (2014) Secure deduplication with efficient and reliable convergent key management. IEEE Trans Parallel Distrib Syst 25(6):1615–1625
55. Wichs D (2011) Cryptographic resilience to continual information leakage. PhD thesis, New York University
56. Naor M, Yung M (1990) Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC 1990, pp 427–437

57. Qin B, Liu S (2013) Leakage-resilient chosen-ciphertext secure public-key encryption from hash proof system and one-time lossy filter. In: Advances in Cryptology-ASIACRYPT 2013. Springer, pp 381–400
58. Naor M, Segev G (2012) Public-key cryptosystems resilient to key leakage. SIAM J Comput 41(4):772–814
59. Cramer R, Shoup V (2002) Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Advances in Cryptology-EUROCRYPT 2002. Springer, pp 45–64
60. Dodis Y, Kalai Y, Lovett S (2009) On cryptography with auxiliary input. In: STOC 2009, pp 621–630
61. Chen J, Wei Lim H, Ling S, Wang H, Wee H (2012) Shorter IBE and signatures via asymmetric pairings. In: Pairing 2012, pp 122–140
62. Yang R, Xu Q, Zhou Y, Zhang R, Hu C, Yu Z (2015) Updatable hash proof system and its applications. In: ESORICS2015, pp 266–285
63. Shoup V (2004) Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive 2004:332

Affiliations

Chengyu Hu[1,2] · Rupeng Yang[3] · Pengtao Liu[4] · Tong Li[5] · Fanyu Kong[6]

✉ Tong Li
litongziyi@mail.nankai.edu.cn

Chengyu Hu
hcy@sdu.edu.cn

Rupeng Yang
orbbyrp@gmail.com

Pengtao Liu
ptwave@163.com

Fanyu Kong
kongfanyu@sdu.edu.cn

1 Software College, Shandong University, Jinan 250101, China

2 Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, Hefei 230601, China

3 School of Computer Science and Technology, Shandong University, Jinan 250101, China

4 School of Information, Shandong University of Political Science and Law, Jinan 250014, China

5 School of Computer Science, Guangzhou University, Guangzhou 510006, China

6 Institute of Network Security, Shandong University, Jinan 250100, China

A Standard Model Secure Verifiably Encrypted Signature Scheme Based on Dual System

Pengtao Liu(✉)

College of Cyberspace Security, Shandong University of Political Science and Law, Jinan, China
ptwave@163.com

Abstract. In this paper, we propose a new standard-model verifiably encrypted signature scheme (VES) based on dual system. The VES scheme is provably secure under the decisional Bilinear Diffie-Hellman and decisional Linear assumptions. Our security proof is based on a newly-introduced strategy called dual system by Waters. In our proof, we divide the signature space into two types and prove that neither type of the signature could be forged or extracted by defining a game sequences in which we can change the query signatures from one type to another one by one without the notice of the adversary.

Keywords: Verifiably encrypted signature · Standard model · Dual system · Opacity · Unforgeability

1 Introduction

Verifiably encrypted signature was first introduced in 1998 by Asokan [1], its first and major usage is in optimistic fair exchange. Following Asokan's pioneering optimistic fair exchange protocol which involve highly intricate interactive ZK proof, lots of VES schemes in the random oracle model have been proposed with only NIZK proof. In 2003, Boneh et al. [2] proposed a non-interactive VES scheme via signature aggregation based on pairings and Zhang et al. [3] proposed a VES scheme derived from their own pairing-based signature [4]. Subsequently, many works have been proposed to construct a VES without random oracle. A standard model secure VES derived from Waters' signature [5] was proposed by Lu et al. [6] in 2006, and in 2007, Zhang et al. [7] proposed another VES scheme which is also derived from Waters' signature [5]. Both VES schemes above have large public parameters. Yet Zhang et al. [7] have not provided a sound security proof. In 2009, Ruckert and Schroder [8] proposed a VES from multilinear maps which has not yet been proved to be constructable. In 2015, Hanser et al. [9] proposed a standard model secure VES scheme from structure-preserving signatures on equivalence classes. Some works also add new attributes to the VES scheme, such as [10] and [11].

Recently, Waters [12] introduced a novel strategy to construct security proof in the standard model. Nishimaki and Xagawa [13] proposed a standard model VES scheme with short keys based on Waters' scheme. In this paper, we also explore Waters' new

J. Vaidya et al. (Eds.): CSS 2019, LNCS 11983, pp. 242–252, 2019.
https://doi.org/10.1007/978-3-030-37352-8_21

strategy and construct a new VES which is fully secure under the decisional Bilinear Diffie-Hellman and decisional Linear assumptions. Our VES scheme is a standard model VES scheme with short system parameters whose signature consists of a constant number of group elements. Specifically, our scheme has shorter verification key compared with that in [13]. In our proof, we have two types of signatures and prove that neither type of the signature could be forged or extracted by defining a sequence of games where we can change the query signatures from one type to another one by one without the notice of the adversary.

The rest of this paper is organized as follows. First, some preliminaries are given in Sect. 2. Then in Sect. 3, we present our VES scheme with its security proof. Section 4 concludes our work.

2 Preliminaries

In this Section, we present some relevant complexity assumptions. In order to save spaces, we omit here the description of bilinear groups [12] and the model of Verifiably Encrypted Signature [13].

2.1 Decisional Bilinear Diffie-Hellman Assumption [12]

Let g be a generator of group G of prime order p. Let $e : G \times G \rightarrow G_T$ be a bilinear map. Select three random elements $c_1, c_2, c_3 \in Z_p$. Given $\vec{T} = (g, g^{c_1}, g^{c_2}, g^{c_3})$, it is hard to distinguish $e(g,g)^{c_1 c_2 c_3}$ from a random element R in G_T.

Let D be an algorithm to distinguish $e(g,g)^{c_1 c_2 c_3}$ from R with advantage ϵ, i.e.,

$$\epsilon = \left| Pr[D(\vec{T}, e(g,g)^{c_1 c_1 c_3}) = 0] \; - \; Pr[D(\vec{T}, R) = 0] \right|$$

Definition 1. We say that the decisional BDH assumption holds if for all polynomial-time algorithms D, ϵ is negligible.

2.2 Decisional Linear Assumption [12]

Let g, f, μ be generators of group G of prime order p. Select two random elements $c_1, c_2 \in Z_p$. Given $\vec{T} = (g, f, \mu, g^{c_1}, f^{c_2})$, it is hard to distinguish $\mu^{c_1 + c_2}$ from a random element R in G.

Let D be an algorithm to distinguish $\mu^{c_1 + c_2}$ from R with advantage ϵ, i.e.,

$$\epsilon = \left| Pr[D(\vec{T}, \mu^{c_1 + c_2}) = 0] \; - \; Pr[D(\vec{T}, R) = 0] \right|$$

Definition 2. We say that the decisional Linear assumption holds if for all polynomial-time algorithms D, ϵ is negligible.

3 Our Scheme

We now present our verifiably encrypted signature scheme along with our proof of its security.

3.1 The Concrete Scheme

VES.Setup. The setup algorithm first chooses groups G, G_T of prime order p, for which there exists an admissible bilinear pairing $e : G \times G \rightarrow G_T$. And choose randomly a generator $g \in G$.

VES.SignKGen. The signer's key generation algorithm chooses randomly generators $v, v_1, v_2, \omega, u, h \in G$ and exponents $a_1, a_2, b, \alpha \in Z_p$. Let $\tau_1 = vv_1^{a_1}$, $\tau_2 = vv_2^{a_2}$. The public key is

$$PK = \left\{g, g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, \omega, u, h, e(g,g)^{\alpha a_1 b}\right\},$$

and the private key is $SK = \{g^\alpha, g^{\alpha a_1}, v, v_1, v_2\}$. The message space is Z_p.

VES.Sign(SK, M). The signature algorithm chooses random $r_1, r_2, z_1, z_2, tag_k \in Z_p$. Let $r = r_1 + r_2$. Then calculate $\sigma_1 = g^{\alpha a_1} v^r$, $\sigma_2 = g^{-\alpha} v_1^r g^{z_1}$, $\sigma_3 = \left(g^b\right)^{-z_1}$, $\sigma_4 = v_2^r g^{z_2}$, $\sigma_5 = \left(g^b\right)^{-z_2}$, $\sigma_6 = g^{r_2 b}$, $\sigma_7 = g^{r_1}$, $\sigma_k = \left(u^M \omega^{tag_k} h\right)^{r_1}$. The signature is

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k).$$

VES.Verify(PK, σ, M). The verification algorithm first chooses random s_1, s_2, t, and $tag_c \in Z_p$ where $tag_c \neq tag_k$. Let $s = s_1 + s_2$. It creates $C_1 = \left(g^b\right)^{s_1+s_2}$, $C_2 = \left(g^{ba_1}\right)^{s_1}$, $C_3 = \left(g^{a_1}\right)^{s_1}$, $C_4 = \left(g^{ba_2}\right)^{s_2}$, $C_5 = \left(g^{a_2}\right)^{s_2}$, $C_6 = \tau_1^{s_1}\tau_2^{s_2}$, $C_7 = \left(\tau_1^b\right)^{s_1}\left(\tau_2^b\right)^{s_2}\omega^{-t}$, $E_1 = \left(u^M \omega^{tag_c} h\right)^t$, $E_2 = g^t$. Then it computes:

$$A_1 = e(C_1, \sigma_1)e(C_2, \sigma_2)e(C_3, \sigma_3)e(C_4, \sigma_4)e(C_5, \sigma_5),\ A_2 = e(C_6, \sigma_6)e(C_7, \sigma_7)$$

Let $A_3 = A_1/A_2$. Then the verification algorithm computes

$$A_4 = \left(e(E_1, \sigma_7)/e(E_2, \sigma_k)\right)^{1/(tag_c - tag_k)}.$$

Finally, it tests the following equation $\left(e(g,g)^{\alpha a_1 b}\right)^{s_2} = A_3/A_4$ and outputs "1" if the equation holds meaning that the VES signature is valid. Otherwise, it outputs "0".

VES.AdjKGen. The adjudicator-key generate algorithm chooses randomly $S_a \in Z_p$, and set public key $APK = g^{S_a}$, private key $ASK = S_a$.

VES.ESign(APK, SK, M). The verifiably encrypted signature algorithm randomly chooses $r_1, r_2, z_1, z_2, tag_k \in Z_p$. Let $r = r_1 + r_2$. Then calculate $\sigma_1 = g^{\alpha a_1} v^r$,

$\sigma_2 = g^{-\alpha} v_1^r g^{z_1}$, $\sigma_3 = \left(g^b\right)^{-z_1}$, $\sigma_4 = v_2^r g^{z_2}$, $\sigma_5 = \left(g^b\right)^{-z_2}$, $\sigma_6 = g^{r_2 b}$, $\sigma_7 = g^{r_1}$, $\sigma_k = (u^M \omega^{tag_k} h \cdot APK)^{r_1}$. The signature is $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$.

VES.Everify(PK, APK, σ, M). The verification algorithm first chooses random s_1, s_2, t, and $tag_c \in Z_p$ where $tag_c \neq tag_k$. Let $s = s_1 + s_2$. It creates $C_1 = \left(g^b\right)^{s_1 + s_2}$, $C_2 = \left(g^{ba_1}\right)^{s_1}$, $C_3 = (g^{a_1})^{s_1}$, $C_4 = \left(g^{ba_2}\right)^{s_2}$, $C_5 = (g^{a_2})^{s_2}$, $C_6 = \tau_1^{s_1} \tau_2^{s_2}$, $C_7 = \left(\tau_1^b\right)^{s_1} \left(\tau_2^b\right)^{s_2} \omega^{-t}$, $E_1 = (u^M \omega^{tag_c} h)^t$, $E_2 = g^t$. Then it computes:

$$A_1 = e(C_1, \sigma_1) e(C_2, \sigma_2) e(C_3, \sigma_3) e(C_4, \sigma_4) e(C_5, \sigma_5),\ A_2 = e(C_6, \sigma_6) e(C_7, \sigma_7)$$

Let $A_3 = A_1 / A_2$. Then the verification algorithm computes $A_4 = (e(E_1, \sigma_7)\, e\left(APK, \sigma_7^t\right) / e(E_2, \sigma_k))^{1/(tag_c - tag_k)}$. Finally, it tests the following equation $\left(e(g,g)^{\alpha a_1 b}\right)^{s_2} = A_3 / A_4$ and outputs "1" if the equation holds meaning that the VES signature is valid. Otherwise, it outputs "0".

VES.Adjudicate(ASK, PK, σ, M). The adjudicate algorithm first verify the validity of σ, if rejected, the algorithm output ⊥. Else, it calculates $\sigma_k' = \sigma_k / \left(\sigma_7^{ASK}\right)$ and keeps other elements unchanged. The algorithm outputs a new signature $\sigma' = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k', tag_k)$.

3.2 Proof of Security

Correctness. The correctness of the VES.Verify algorithm is obviously trivial. The correctness of the VES.Adjudicate algorithm follows from the fact that all the temporary values calculated out in both the VES.Verify and the VES.EVerify algorithms are equal. Given a valid verifiably encrypted signature σ, it satisfies that

$$\begin{aligned} A_4 &= \left(e(E_1, \sigma_7) e\left(APK, \sigma_7^t\right) / e(E_2, \sigma_k)\right)^{1/(tag_c - tag_k)} \\ &= \left(e(E_1, \sigma_7) / e\left(E_2, \sigma_k'\right)\right)^{1/(tag_c - tag_k)} \end{aligned}$$

So, if a signature σ is accepted by VES.EVerify algorithm, then its adjudicated version σ' would definitely be accepted by VES.Verify algorithm.

Unforgeability. The Unforgeability of this VES scheme comes handily from the unforgeability of Waters' dual system signature scheme.

Theorem 1. If Waters' dual system signature scheme is unforgeable, then our verifiably encrypted signature scheme is unforgeable.

Proof. Assume that A is an adversary who can $\left(t, q_{es}, q_{adj}, \epsilon\right)$-break the unforgeability of our scheme, then we can construct an algorithm B which can (t', q_s, ϵ)-break the unforgeability of Waters' dual system signature scheme, where $t' = t + O\left(q_{es} + q_{adj}\right)$ and $q_s = q_{es}$.

When playing the unforgeability game of waters' original dual system signature scheme, algorithm B is given all the public parameters, then B takes these public parameters as the public parameters in our scheme and runs the VES.AdjKGen to get adjudicator's key (ASK, APK). B give all the public parameters and adjudicator's public key APK to the adversary A.

- VES.ESign query. When the adversary A makes VES.ESign queries, B forwards the query parameters to the signing oracle in waters' game, and get a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$, then B set $\sigma_k^{'} = \sigma_k \sigma_7^{ASK}$, and output a verifiably encrypted signature $\sigma' = \left(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k^{'}, tag_k\right)$.
- VES.Adjudicate query. When the adversary A makes VES.Adjudicate queries, B runs the VES.Adjudicate algorithm and output its result.

Output: Finally, the adversary A forges a VES signature σ^* on a message M^*, then the algorithm B could run the VES.Adjudicate algorithm on σ^* and M^*, and get a forged normal signature on M^* which has not been submitted to the signing oracle in Waters' game. Hence the algorithm B could break the unforgeability of the Waters' dual system signature scheme.

Opacity. To prove the opacity security, we partition the whole normal signature space into two different types and argue that an adversary could not extract a normal signature of either type. First, we define a new verification algorithm and divide the normal signature space into two part using the new verification algorithm.

VES.SemiVerify(PK, σ, M). The semi-verification algorithm first chooses randomly x, s_1, s_2, t, and $tag_c \in Z_p$ where $tag_c \neq tag_k$. Let $s = s_1 + s_2$. It creates $C_1 = \left(g^b\right)^{s_1+s_2}$, $C_2 = \left(g^{ba_1}\right)^{s_1}$, $C_3 = (g^{a_1})^{s_1}$, $C_4 = \left(g^{ba_2}\right)^{s_2} g^{ba_2 x}$, $C_5 = (g^{a_2})^{s_2} g^{a_2 x}$, $C_6 = \tau_1^{s_1} \tau_2^{s_2} v_2^{a_2 x}$, $C_7 = \left(\tau_1^b\right)^{s_1} \left(\tau_2^b\right)^{s_2} \omega^{-t} v_2^{a_2 bx}$, $E_1 = (u^M \omega^{tag_c} h)^t$, $E_2 = g^t$. Then it computes

$$A_1 = e(C_1, \sigma_1) e(C_2, \sigma_2) e(C_3, \sigma_3) e(C_4, \sigma_4) e(C_5, \sigma_5),\ A_2 = e(C_6, \sigma_6) e(C_7, \sigma_7)$$

Let $A_3 = A_1 / A_2$. Then the verification algorithm computes $A_4 = (e(E_1, \sigma_7) / e(E_2, \sigma_k))^{1/(tag_c - tag_k)}$. Finally, it tests the following equation

$$\left(e(g,g)^{\alpha a_1 b}\right)^{s_2} = A_3 / A_4$$

and outputs "1" if the equation holds meaning that the VES signature is valid. Otherwise, it outputs "0". Notice that to run VES.SemiVerify, a secret $v_2^{a_2 b}$ is needed. Together with VES.Verify, we can divide normal signatures into two types.

1. **Type A* normal signatures.** The signature that could be accepted by both VES. Verify and VES.SemiVerify with significant probability.
2. **Type B* normal signatures.** The signature that could be accepted by VES.Verify with significant probability and be accepted by VES.SemiVerify with negligible probability.

And we define two concrete types of signatures which could be used in our proof.

1. **Type A normal signature** and **type A verifiably encrypted signature**
   As defined in the VES.Sign and VES.ESign algorithm. It could be easily verified that a type A normal signature is also a type A* normal signature and has 100% probability of being accepted by verify algorithms.
2. **Type B normal signature** and **type B verifiably encrypted signature**
   Taken a type A normal signature or a type A verifiably encrypted signature σ, choose randomly a value $\gamma \in Z_p$ and compute $\sigma_1^{'} = \sigma_1 g^{-a_1 a_2 \gamma}$, $\sigma_2^{'} = \sigma_2 g^{a_2 \gamma}$, $\sigma_4^{'} = \sigma_4 g^{a_1 \gamma}$. The type B signature is $\sigma' = \left(\sigma_1^{'}, \sigma_2^{'}, \sigma_3, \sigma_4^{'}, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k\right)$. It could be easily verified that a type B normal signature is also a type B* normal signature and has 100% probability of being accepted by VES.Verify.

And we define a new game here

Game(i,j): Similar to the opacity game defined in Sect. 2, except that

1. The first i VES.ESign queries always yield type B verifiably encrypted signature and the other VES.ESign queries always yield type A verifiably encrypted signature.
2. The first j VES.Adjudicate queries always yield type B normal signature and the other VES.Adjudicate queries always yield type A normal signature.

Notice that Game$(0, 0)$ is the real opacity game.

Lemma 1. No adversary could produce type B* normal signature in game$(0, 0)$, where the VES.ESign gives out only type A verifiably encrypted signature and VES.Adjudicate gives out only type A normal signature.

Proof. Assume that there exists an algorithm A that makes at most q_s VES.ESign queries and q_{adj} VES.Adjudicate queries and has non-negligible chance of ϵ to forge a type B* normal signature which would be accepted by verify algorithms with non-negligible probability ε in game$(0, 0)$, then we can build an algorithm B that has advantage $\epsilon \cdot \varepsilon$ in the decision linear game. Given $(g, f, \mu, g^{c_1}, f^{c_2}, T)$, the algorithm B wants to decide whether T is equal to $\mu^{c_1 + c_2}$ or T is just a random element in G.

Setup. The algorithm B chooses randomly $b, \alpha, y_v, y_{v1}, y_{v2} \in Z_p$ and $u, \omega, h, Q \in G$. It then sets $g^{a_1} = f, g^{a_2} = \mu$, $g^{ba_1} = f^b, g^{ba_2} = \mu^b, v = g^{y_v}, v_1 = g^{y_{v1}}, v_2 = g^{y_{v2}}$. And calculate $\tau_1, \tau_2, \tau_1^b, \tau_2^b$ and $e(g, g)^{aa_1 b} = e(g, f)^{ab}$ in order to publish the public key PK, and the adjudicator's public key $APK = Q$.

VES.ESign. Since the algorithm B has the secret key SK, it can run the VES.ESign and output the signature.

VES.Adjudicate. Since the algorithm B has the secret key SK, it can run the VES. Sign and output the signature.

Output. The algorithm A outputs a type B* normal signature $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$, then B random choose $s_1, s_2, t, tag_c \in Z_p$,

where $\text{tag}_c \neq \text{tag}_k$ set $s = s_1 + s_2$ and calculate $C_1 = \left(g^b\right)^{s_1+s_2}(g^{c_1})^b$, $C_2 = \left(g^{ba_1}\right)^{s_1}(f^{c_2})^{-b}$,$C_3 = (g^{a_1})^{s_1}f^{-c_2}$, $C_4 = \left(g^{ba_2}\right)^{s_2}T^b$, $C_5 = (g^{a_2})^{s_2}T$, $C_6 = \tau_1^{s_1}\tau_2^{s_2}(g^{c_1})^{y_v}(f^{c_2})^{-y_{v1}}T^{y_{v2}}$, $C_7 = \left(\tau_1^b\right)^{s_1}\left(\tau_2^b\right)^{s_2}\omega^{-t}((g^{c_1})^{y_v}(f^{c_2})^{-y_{v1}}T^{y_{v2}})^b$, $E_1 = (u^M\omega^{tag_c}h)^t$, $E_2 = g^t$; $A_1 = e(C_1,\sigma_1)e(C_2,\sigma_2)e(C_3,\sigma_3)e(C_4,\sigma_4)e(C_5,\sigma_5)$, $A_2 = e(C_6,\sigma_6)e(C_7,\sigma_7)$, $A_3 = A_1/A_2$, $A_4 = (e(E_1,\sigma_7)/e(E_2,\sigma_k))^{1/(tag_c - tag_k)}$. If the equation $\left(e(g,g)^{\alpha a_1 b}\right)^{s_2}(e(g^{c_1},f)e(g,f^{c_2}))^{ba} = A_3/A_4$ holds, then T is equal to $\mu^{c_1+c_2}$ and the construction above is the same as in VES.Verify algorithm, else T is random and the construction above is the same as in VES.SemiVerify algorithm, and algorithm B solve the decision linear problem. Notice that the construction above is either a VES.Verify or a VES.SemiVerify with the implicit value $s_1' = s_1 - c_2$ and $s_2' = s_2 + c_1 + c_2$.

Lemma 2. No adversary could distinguish game$(k, 0)$ from game$(k+1, 0)$.

Proof. Assume that there exists an algorithm A that makes at most q_{es} VES.ESign queries and q_{adj} VES.Adjudicate queries and has non-negligible chance of ϵ to distinguish game$(k, 0)$ from game$(k+1, 0)$, then we can build an algorithm B that has advantage ϵ in the decision linear game. Given $(g, f, \mu, g^{c_1}, f^{c_2}, T)$, the algorithm B wants to decide whether T is equal to $\mu^{c_1+c_2}$ or T is just a random element in G.

Setup. Algorithm B first chooses randomly $\alpha, a_1, a_2, y_{v1}, y_{v2}, y_\omega, y_u, y_h$. And it sets $g^b = f, g^{ba_1} = f^{a_1}, g^{ba_2} = f^{a_2}, v = \mu^{-a_1a_2}, v_1 = \mu^{a_2}g^{y_{v1}}, v_2 = \mu^{a_1}g^{y_{v2}}, e(g,g)^{\alpha a_1 b} = e(f,g)^{\alpha a_1}$. Then algorithm B can create $\tau_1 = vv_1^{a_1} = g^{y_{v1}a_1}, \tau_2 = vv_2^{a_2} = g^{y_{v2}a_2}$, $\tau_1^b = f^{y_{v1}a_1}, \tau_2^b = f^{y_{v2}a_2}$. Finally, B chooses randomly $s, A, B \in Z_p$ and sets $\omega = fg^{y_\omega}, u = f^{-A}g^{y_u}, h = f^{-B}g^{y_h}, Q = g^s$ in order to publish all the public parameters of the system PK and the adjudicator's public key $APK = Q$. Note that although B knows the secret key of the adjudicator here, the importance here is not to break the opacity but to distinguish the two games.

VES.ESign. We break the VES.ESign queries into three cases. Consider the i-th query made by A.

Case 1: $i > k+1$. Since B has the secret key SK, it could run the VES.ESign to produce a type A verifiably encrypted signature.

Case 2: $i < k+1$. Since B has the secret key SK and g^{a1a2}, it could produce a type B verifiably encrypted signature.

Case 3: $i = k+1$. B first runs the VES.ESign to generate a type A verifiably encrypted signature $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$ where $tag_k = A * M + B$, let r_1, r_2, z_1, z_2 be the random exponents used in VES.ESign algorithm. It then sets $\sigma_1' = \sigma_1 T^{-a_1a_2}$, $\sigma_2' = \sigma_2 T^{a_2}(g^{c_1})^{y_{v1}}$, $\sigma_3' = \sigma_3(f^{c_2})^{y_{v1}}$, $\sigma_4' = \sigma_4 T^{a_1}(g^{c_1})^{y_{v2}}$, $\sigma_5' = \sigma_5(f^{c_2})^{y_{v2}}$, $\sigma_6' = \sigma_6 f^{c_2}$, $\sigma_7' = \sigma_7(g^{c_1})$, $\sigma_k' = \sigma_k(g^{c_1})^{M*y_u + y_h + tag_k*y_\omega + s}$, and return the new signature $M, \sigma' = \left(\sigma_1', \sigma_2', \sigma_3', \sigma_4', \sigma_5', \sigma_6', \sigma_7', \sigma_k', tag_k\right)$. If T is equal to $\mu^{c_1+c_2}$, then this new signature is a type A verifiably encrypted signature under randomness $r_1' = r_1 + c_1$ and $r_2' = r_2 + c_2$. Otherwise, if T is random, then we can write $T = \mu^{c_1+c_2}g^r$ for random $r \in Z_p$. This is a type B verifiably encrypted signature where r is the added randomness.

VES.Adjudicate. Since B has the secret key SK, it could run the VES.Sign algorithm to produce a type A normal signature.

Output. If algorithm A decides that the game constructed above is game$(k, 0)$, then B could claim that T is equal to $\mu^{c_1+c_2}$, otherwise B could claim that T is a random group element.

Lemma 3. If an adversary could produce a type A* normal signature in game$(k, 0)$, then it could also produce a type A* normal signature in game$(k+1, 0)$.

Proof. If there is an adversary who can produce a type A* normal signature in game $(k, 0)$ but could not produce any normal signature in game$(k+1, 0)$, we can easily construct a algorithm to distinguish game$(k, 0)$ from game$(k+1, 0)$ and break Lemma 2 above. So we only need to prove that under this condition the adversary could not produce a type B* normal signature in game$(k+1, 0)$.

Assume that the adversary A has non-negligible chance of ϵ to produce a type A* normal signature which would be accepted by verifying algorithms with non-negligible probability ε in game$(k, 0)$ and had non-negligible chance of ϵ' to produce a type B* normal signature which would be accepted by veirfing algorithms with non-negligible probability ε' in game$(k+1, 0)$, then we can build an algorithm B that has advantage $\epsilon \cdot \varepsilon \cdot \epsilon' \cdot \varepsilon'$ in the decision linear game. Given $(g, f, \mu, g^{c_1}, f^{c_2}, T)$, the algorithm B wants to decide whether T is equal to $\mu^{c_1+c_2}$ or T is just a random element in G.

The constructions of Setup, VES.ESign and VES.Adjudicate are the same as that in the proof of Lemma 2.

Output. The algorithm A output a type B* normal signature $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$. Since algorithm A has no knowledge of the value of A and B, we claim that it has negligible chance to produce a signature with $tag_k = A * M + B$. Assuming $tag_k \neq A * M + B$, B randomly chooses $x, s_1, s_2, t \in Z_p$. Set $s = s_1 + s_2$, $tag_c = A * M + B$ and calculate $C_1 = \left(g^b\right)^{s_1+s_2}$, $C_2 = \left(g^{ba_1}\right)^{s_1}$, $C_3 = (g^{a_1})^{s_1}$, $C_4 = \left(g^{ba_2}\right)^{s_2} f^{a_2 x}$, $C_5 = (g^{a_2})^{s_2} g^{a_2 x}$, $C_6 = \tau_1^{s_1} \tau_2^{s_2} v_2^{a_2 x}$, $C_7 = \left(\tau_1^b\right)^{s_1} \left(\tau_2^b\right)^{s_2} \omega^{-t} f^{y_{v2} \cdot x \cdot a_2} \mu^{-a_1 \cdot x \cdot y_\omega \cdot a_2}$, $E_1 = (u^M \omega^{tag_c} h)^t (\mu^{My_u + y_h + tag_c y_\omega})^{a_1 a_2 x}$, $E_2 = g^t \mu^{a_1 a_2 x}$. Then B computes:

$A_1 = e(C_1, \sigma_1) e(C_2, \sigma_2) e(C_3, \sigma_3) e(C_4, \sigma_4) e(C_5, \sigma_5)$, $A_2 = e(C_6, \sigma_6) e(C_7, \sigma_7)$, $A_3 = A_1 / A_2$ and $A_4 = (e(E_1, \sigma_7) / e(E_2, \sigma_k))^{1/(tag_c - tag_k)}$. If the following equation $\left(e(g, g)^{\alpha a_1 b}\right)^{s_2} = A_3 / A_4$ holds, the algorithm B could decide that the signature produced by A is a type A* normal signature, otherwise the signature is a type B* normal signature. In either case, B could distinguish game$(k, 0)$ from game$(k+1, 0)$ and break the decision linear problem. Notice that the construction above is essentially the same as in VES.SemiVerify algorithm which could distinguish a type A* normal signature from a type B* normal signature with significant probability.

Lemma 4. No adversary could distinguish game(q_{es}, k) from game$(q_{es}, k+1)$

Proof. Assume that there exists an algorithm A that makes at most q_{es} VES.ESign queries and q_{adj} VES.Adjudicate queries and has non-negligible chance of ϵ to distinguish game(q_{es}, k) from game$(q_{es}, k+1)$, then we can build an algorithm B that has

advantage ϵ in the decision linear game. Given $(g, f, \mu, g^{c_1}, f^{c_2}, T)$, the algorithm B wants to decide whether T is equal to $\mu^{c_1+c_2}$ or T is random in G.

Setup. Algorithm B first chooses randomly $\alpha, a_1, a_2, y_{v1}, y_{v2}, y_\omega, y_u, y_h$. And it sets $g^b = f, g^{ba_1} = f^{a_1}, g^{ba_2} = f^{a_2}, v = \mu^{-a_1 a_2}, v_1 = \mu^{a_2} g^{y_{v1}}, v_2 = \mu^{a_1} g^{y_{v2}}, e(g,g)^{\alpha a_1 b} = e(f,g)^{\alpha a_1}$. Then algorithm B can create $\tau_1 = v v_1^{a_1} = g^{y_{v1} a_1}, \tau_2 = v v_2^{a_2} = g^{y_{v2} a_2}$, $\tau_1^b = f^{y_{v1} a_1}, \tau_2^b = f^{y_{v2} a_2}$. At last, B chooses randomly $s, A, B \in Z_p$ and sets $\omega = f g^{y_\omega}, u = f^{-A} g^{y_u}, h = f^{-B} g^{y_h}, Q = g^s$ to publish all the public parameters of the system PK and the adjudicator's public key $APK = Q$. Note that although B knows the secret key of the adjudicator here, the importance here is not to break the opacity but to distinguish the two games.

VES.ESign Query. Since B has the secret key SK and $g^{a_1 a_2}$, it could run the VES. ESign algorithm to generate a type A VES signature and apply the algorithm in the definition of type B VES signature to generate a type B VES signature.

VES.Adjudicate Query. We break the VES.Adjudicate queries into three cases. Consider the i-th query made by A.

Case 1: $i > k+1$. Since B has the secret key SK, it could run the VES.Sign to produce a type A normal signature.
Case 2: $i < k+1$. Since B has the secret key SK and $g^{a_1 a_2}$, it could produce a type B normal signature.
Case 3: $i = k+1$. B first runs the VES.Sign to generate a type A normal signature $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$ where $tag_k = A * M + B$, let r_1, r_2, z_1, z_2 be the random exponents used in VES.Sign algorithm. It then sets $\sigma_1^{'} = \sigma_1 T^{-a_1 a_2}$, $\sigma_2^{'} = \sigma_2 T^{a_2} (g^{c_1})^{y_{v1}}$, $\sigma_3^{'} = \sigma_3 (f^{c_2})^{y_{v1}}$, $\sigma_4^{'} = \sigma_4 T^{a_1} (g^{c_1})^{y_{v2}}$, $\sigma_5^{'} = \sigma_5 (f^{c_2})^{y_{v2}}$, $\sigma_6^{'} = \sigma_6 f^{c_2}$, $\sigma_7^{'} = \sigma_7 (g^{c_1})$, $\sigma_k^{'} = \sigma_k (g^{c_1})^{M * y_u + y_h + tag_k * y_\omega}$, and return the new signature $M, \sigma' = \left(\sigma_1^{'}, \sigma_2^{'}, \sigma_3^{'}, \sigma_4^{'}, \sigma_5^{'}, \sigma_6^{'}, \sigma_7^{'}, \sigma_k^{'}, tag_k\right)$. If T is equal to $\mu^{c_1+c_2}$, then this new signature is a type A normal signature under randomnesses $r_1^{'} = r_1 + c_1$ and $r_2^{'} = r_2 + c_2$. Otherwise, if T is random, then it can be written as $T = \mu^{c_1+c_2} g^r$ for random $r \in Z_p$. This is a type B normal signature where r is the added randomness.

Output. If algorithm A decides that the game above is game(q_{es}, k), then B could claim that T is equal to $\mu^{c_1+c_2}$, otherwise B could claim that T is a random element.

Lemma 5. If an adversary could produce a type A* normal signature in game(q_{es}, k), then it could also produce a type A* normal signature in game$(q_{es}, k+1)$.

Proof. Similar proof sketch as in Lemma 3.

Lemma 6. No adversary could produce a type A* normal signature in game(q_{es}, q_{adj}), where the oracles give out only type B normal signatures and verifiably encrypted signatures.

Proof. Assume that there exists an algorithm A that makes at most q_{es} VES.ESign queries and q_{adj} VES.Adjudicate queries and has non-negligible chance of ϵ to forge a new type A* normal signature which would be accepted by veirfy algorithms with

non-negligible probability ε, then we can build an algorithm B that has advantage $\varepsilon \cdot \epsilon$ in the BDH game. Given a BDH instance $(g, g^{c1}, g^{c2}, g^{c3}, T)$, algorithm B wants to decide whether T is equal to $e(g,g)^{c_1c_2c_3}$.

Setup. The algorithm B first chooses randomly $a_1, b, y_v, y_{v1}, y_{v2}, y_\omega, y_h, y_u \in Zp$, $Q \in G_1$. It then sets $g^b, g^{a_1}, g^{a_2} = g^{c_2}, g^{ba_1}, g^{ba_2} = (g^{c_2})^b$, $v = g^{y_v}, v_1 = g^{y_{v1}}, v_2 = g^{y_{v2}}, \omega = g^{y_\omega}, u = g^{y_u}, h = g^{y_h}, e(g,g)^{a_1 \alpha b} = e(g^{c_1}, g^{c_2})^{a_1 b}$ and calculates $\tau_1 = v v_1^{a_1}$, $\tau_1^b, \tau_2 = v(g^{c_2})^{y_{v2}}, \tau_2^b$ in order to public the public key PK and $APK = Q$. Notice that B dose not know the secret key SK.

VES.ESign. Given a message M, B randomly chooses $r_1, r_2, z_1, z_2, \gamma, tag_k \in Z_p$ and defines $r = r_1 + r_2$. It creates a signature as: $\sigma_1 = (g^{c_2})^{-\gamma a_1} v^r$, $\sigma_2 = (g^{c_2})^\gamma v_1^r g^{z_1}$, $\sigma_3 = \left(g^b\right)^{-z_1}$, $\sigma_4 = (g^{c_1})^{a_1} g^{a_1 \gamma} v_2^r g^{z_2}$, $\sigma_5 = \left(g^b\right)^{-z_2}$, $\sigma_6 = g^{r_2 b}$, $\sigma_7 = g^{r_1}$, $\sigma_k = (u^M \omega^{tag_k} hQ)^{r_1}$ and outputs the signature as $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$.

VES.Adjudicate. Given a message M, the algorithm B randomly chooses $r_1, r_2, z_1, z_2, \gamma, tag_k \in Z_p$ and defines $r = r_1 + r_2$. It creates a signature as: $\sigma_1 = (g^{c_2})^{-\gamma a_1} v^r$, $\sigma_2 = (g^{c_2})^\gamma v_1^r g^{z_1}$, $\sigma_3 = \left(g^b\right)^{-z_1}$, $\sigma_4 = (g^{c_1})^{a_1} g^{a_1 \gamma} v_2^r g^{z_2}$, $\sigma_5 = \left(g^b\right)^{-z_2}$, $\sigma_6 = g^{r_2 b}$, $\sigma_7 = g^{r_1}$, $\sigma_k = (u^M \omega^{tag_k} h)^{r_1}$ and outputs the signature as $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$.

Output. Algorithm A output a type A* normal signature $M, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_k, tag_k)$, algorithm B could use this signature to solve the BDH problem. B randomly chooses $s_1, t, tag_c, x \in Z_p$ and calculates $C_1 = g^{s_1 b}(g^{c_3})^b$, $C_2 = g^{ba_1 s_1}$, $C_3 = g^{a_1 s_1}$, $C_4 = (g^{c_2})^{xb}$, $C_5 = (g^{c_2})^x$, $C_6 = \tau_1^{s_1}(g^{c_3})^{y_v}$ $(g^{c_2})^{y_{v2} x}$, $C_7 = \left(\tau_1^b\right)^{s_1}(g^{c_3})^{y_v b}$ $(g^{c_2})^{y_{v2} xb} \omega^{-t}$, $E_1 = (u^M \omega^{tag_c} h)^t$, $E_2 = g^t$; $A_1 = e(C_1, \sigma_1)$ $e(C_2, \sigma_2) e(C_3, \sigma_3) e(C_4, \sigma_4)$ $e(C_5, \sigma_5)$, $A_2 = e(C_6, \sigma_6) e(C_7, \sigma_7)$, $A_3 = A_1 / A_2$, $A_4 = (e(E_1, \sigma_7)/e(E_2, \sigma_k))^{1/(tag_c - tag_k)}$. If the equation $T^{a_1 b} = A_3 / A_4$ holds, then T is equal to $e(g,g)^{c_1 c_2 c_3}$ or else T is just random.

Lemma 7. No adversary could produce a type A* normal signature in game$(0,0)$, where the oracles give out only type A normal signatures and verifiably encrypted signatures.

Proof. Directly from Lemmas 3, 5 and 6.

Theorem 2. No adversary should break the opacity game.

Proof. Directly from Lemmas 1 and 7.

4 Conclusion

In this paper, we construct another verifiably encrypted signature scheme in standard model based on Waters' newly introduced Dual System and give its security analysis. Compared with other standard model verifiably encrypted signature schemes, our

construction has const number of public parameters and uses simple assumptions for security analysis. The drawback is that the signature size is somewhat larger than in other system.

References

1. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054156
2. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_26
3. Zhang, F., Safavi-Naini, R., Susilo, W.: Efficient verifiably encrypted signature and partially blind signature from bilinear pairings. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 191–204. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24582-7_14
4. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24632-9_20
5. Waters, B.: Efficient identity-based encryption without random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_7
6. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_28
7. Zhang, J., Mao, J.: A novel verifiably encrypted signature scheme without random Oracle. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 65–78. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72163-5_7
8. Rückert, M., Schröder, D.: Aggregate and verifiably encrypted signatures from multilinear maps without random Oracles. In: Park, J.H., Chen, H.-H., Atiquzzaman, M., Lee, C., Kim, T.-h., Yeo, S.-S. (eds.) ISA 2009. LNCS, vol. 5576, pp. 750–759. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02617-1_76
9. Hanser, C., Rabkin, M., Schröder, D.: Verifiably encrypted signatures: security revisited and a new construction. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 146–164. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_8
10. Wang, Y., Pang, H., Deng, R.H.: Verifiably encrypted cascade-instantiable blank signatures to secure progressive decision management. Int. J. Inf. Secur. **17**, 347–363 (2018)
11. Wang, Z., Luo, X., Wu, Q.: Verifiably encrypted group signatures. In: Okamoto, T., Yu, Y., Au, M.H., Li, Y. (eds.) ProvSec 2017. LNCS, vol. 10592, pp. 107–126. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68637-0_7
12. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36
13. Nishimaki, R., Xagawa, K.: Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted VES. Des. Codes Crypt. **77**, 61–98 (2015)

Received: 10 August 2022 | Revised: 16 October 2022 | Accepted: 2 December 2022

DOI: 10.1002/ett.4712

RESEARCH ARTICLE

WILEY

Blockchain-based secure deduplication of encrypted data supporting client-side semantically secure encryption without trusted third party

Guiyun Qin[1] | **Limin Li**[1] | **Pengtao Liu**[2] | **Chengyu Hu**[1,3,4] | **Shanqing Guo**[1,3,4]

[1]School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China

[2]School of Cyberspace Security, Shandong University of Political Science and Law, Jinan, Shandong, China

[3]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, Shandong, China

[4]Quancheng Laboratory, Jinan, Shandong, China

Correspondence
Chengyu Hu, School of Cyber Science and Technology, Shandong University, 72 Binhai Rd, Qingdao, China.
Email: hcy@sdu.edu.cn

Funding information
Shandong Provincial Natural Science Foundation, Grant/Award Numbers: ZR2020LZH002, ZR2020MF055, ZR2021LZH007, ZR2020QF045; The Open Project of Key Laboratory of Network Assessment Technology, Institute of Information Engineering, Chinese Academy of Sciences, Grant/Award Number: KFKT2019-002

Abstract

To accommodate the new demand for the deduplication of encrypted data, secure encrypted data deduplication technologies have been widely adopted by cloud service providers. At present, of particular concern is how deduplication can be applied to the ciphertexts encrypted by semantically secure symmetric encryption scheme. Avoiding disadvantages of the existing methods, in this article, we propose a blockchain-based secure encrypted data deduplication protocol supporting client-side semantically secure encryption. In the proposed protocol, the smart contracts are deployed by the first file uploader, and then the subsequent uploaders implement an interactive proof of ownership for the same file with the help of the smart contracts executing a cloud data integrity auditing protocol. The smart contracts play the role of the trusted third party and therefore make up for the poor feasibility for the existence of a trusted third party in real scenario. In addition, in the proposed protocol, there is no need for other clients who have uploaded the same file to be online to help the current uploader obtain the encryption key. We also prove its security and evaluate its performance.

1 | INTRODUCTION

The development of cloud storage technology prompts the demand for outsourcing massive data to the cloud servers. Cloud storage brings obvious benefits to data owners, such as saving users from the management and maintenance of local storage, saving local resources, and allowing users to use any device such as smartphones, laptops, and desktop computers anytime and anywhere to access the outsourced data in the cloud server, regardless of time and location. Subsequently, cloud storage brings many problems. In actual applications, cloud service providers are not completely trusted, and data leakage often occurs. For example, Facebook disclosed the user's contact information in 2013;[1] Icloud disclosed users' private photos in 2014;[2] Cathay Pacific Airways Limited leaked passenger personal information, including the passenger's name, passport, and identity information, telephone number, etc., in 2019.[3] To ensure data privacy,

users usually encrypt data before uploading it to the cloud server. As a result, the same data is encrypted into different ciphertexts and stored repeatedly by different users, resulting in a huge waste of storage space. According to the "Data Age 2025" whitepaper of International Data Corporation (IDC), the global data volume is expected to reach 175 Zettabytes by 2025.[4] Another IDC survey shows that 75% of the data are duplicate.[5] Because redundant data wastes an abundance of storage resources, researchers have proposed many different data deduplication protocols based on different encryption algorithms. However, the various deduplication protocols currently on the market have also been proven to have a large number of security risks and inefficiencies: the protocols based on convergent encryption are vulnerable to offline brute-force attacks; the requirement that there exists a trusted third party for some protocols is difficult to meet in reality; the protocols supporting semantically secure client-side encryption without the need for additional independent servers may be suffering some additional attacks or overheads as a result of the requirement for other clients who have uploaded the same encrypted file to implement a key distribution protocol with the current uploader. Therefore, how to securely and efficiently delete duplicate encrypted data[6] has become one of the current research hotspots in the field of cloud storage security.

1.1 | Related work

To protect data privacy, the researchers proposed some methods to deduplicate encrypted data stored on the cloud server. The client uses convergent encryption (CE)[7] and message locked encryption (MLE) scheme[8] to encrypt data, which provides users with practical data privacy options while using the deduplication function. CE was proposed by Douceur et al[7] which computes and uses the hash value of the data as the encryption key, and ensures that the same data always corresponds to the same ciphertext. Although CE is efficient, it does not achieve semantic security and is vulnerable to offline brute-force attacks.[9] Bellare et al proposed a MLE scheme encrypting data under the MLE key which comes from the encrypted hash value of data. In this way, MLE ensures that the same data will be encrypted into the same ciphertext. Compared with CE, the core idea of MLE has not changed so that it is also unable to achieve semantic security.[10,11] Bellare et al[12] proposed *DupLESS*, in which different owners of the same data and the trusted key server execute the obfuscated pseudorandom function (OPF) to generate the same encryption key. To improve the security, Duan[13] proposed a scheme in which the key server is eliminated and a distributed key generation protocol is implemented by the clients before uploading the data. Both Bellare et al's and Duan's protocols[12,13] cannot resist the online exhaustive attack of cloud servers. Puzio et al[14] put forward the first double-encryption based data deduplication protocol ClouDedup in which the inner layer uses efficient CE and the outer layer outsources the encryption and decryption work to a trusted third party. Although the security is improved, double-layer encryption technology brings high computing and communication costs. In addition, CloudeDup cannot prevent collusion attacks between cloud service providers and the third parties. Recently, three encrypted data deduplication methods achieving semantic security[15-17] have been proposed. However, these methods need a trusted third party to generate for each user some auxiliary information such as the key for broadcast encryption (BE)/attribute-based encryption (ABE) and parameters encrypted by BE/ABE scheme. This makes that each user can obtain parameters of other users. Therefore these methods cannot resist collision attacks, that is, if the cloud server compromises one or more users, it can obtain the data of other users. As is known to all, in a real scenario, it is difficult to deploy a fully trusted third party. Therefore, Liu et al designed the protocols[9,18] of encrypted data deduplication without the participation of the trusted third party, using password authenticated key exchange (PAKE) to transfer the encryption key. The server requires the client with the same short hash value to execute the PAKE protocol with the file hash value as the input password, and finally output the session key. After that, clients with the same data will get the same session key. Through this key, a subsequent uploader of the same data can get the encryption key used by the previous uploader. Although the protocol can achieve semantic security, the requirement that some participants must be online leads to less practical.

1.2 | Our contribution

As we mentioned above, existing secure deduplication protocols of encrypted data supporting client-side semantically secure encryption without trusted third-party[9,18] require some clients who have uploaded the same data to be online and run PAKE protocol when a client wants to upload data. To solve this problem, in this article, we propose a

blockchain-based secure deduplication protocol of encrypted data. Based on the advantages of blockchain,[19] our protocol supports client-side semantically secure encryption without the trusted third party and does not need other uploaders to be online. Different from the traditional credit endorsement mechanism relying on the trusted third party, blockchain technology is deeply integrated through P2P network, cryptography technology, consensus mechanism, etc., to solve the problem of establishing a trust mechanism between nodes in a decentralized system.[20,21] Therefore, based on the blockchain, our deduplication protocol of encrypted data solves the third-party trust mechanism problem. In our protocol, smart contracts are used to implement the proof of data ownership. Before uploading the file, we first check whether the file already exists in the cloud server. That is, the current uploader uploads the short hash value of the file to the cloud server to find whether there is an equivalent short hash value in the cloud server. If it exists, the cloud server sends the address set of smart contracts T_0 and T_1 deployed by the original uploader to the current uploader and the current uploader executes the smart contracts. If it is confirmed that the current uploader does fully own the file F, he can obtain the encryption key used by the original uploader. Otherwise, the current uploader is the first uploader. Then he sends the ciphertext of the file and the addresses of the constructed smart contracts T_0 and T_1 to the server. In this way, our protocol does not require other uploaders to be online.

We proved the security of the protocol in the adversarial model. We also analyzed the performance of our protocol. We implemented three experiments by controlling the data size, the block size, and the number of selected checked blocks to study the durations of tag, challenge and proof generation. In addition, it is that our solution does not need to call other uploaders to participate in the protocol, that prevents the collusion attacks of malicious uploaders and the cloud server, and improves the security. On the whole, compared with the existing encrypted data deduplication protocols, our protocol is much more practical.

1.3 | Organization

The rest of this article is organized as follows: Section 2 introduces some preliminaries of our work and gives an overview of our solution. Section 3 describes our blockchain-based deduplication protocol in detail and analyzes the security of the protocol. In Section 4, some experiments are implemented to evaluate the performance of our protocol. Some privacy-preserving technologies for the data encryption key stored in the blockchain are discussed in Section 5. Finally, Section 6 concludes the article.

2 | PRELIMINARIES

In this section, we describe some preliminaries.

2.1 | Cloud data integrity auditing protocol

A cloud data integrity auditing protocol consists of the following four algorithms (SysSetup, AuthGen, ProofGen, ProofVerify):[22]

- SysSetup$(1^\lambda)\rightarrow (PK, SK)$: The system setup algorithm is executed by the client and takes as input a security parameter λ, and generates a public key PK and the client's secret key SK.
- AuthGen$(PK, SK, F)\rightarrow (\Phi)$: The authenticator generation algorithm is executed by the client and takes as input the public key PK, the client's current secret key SK and a file F, and generates the set of authenticators Φ for F.
- ProofGen$(PK, Chal, F, \Phi)\rightarrow (P)$: The proof generation algorithm is run by the cloud server and takes as input the public key PK, a challenge $Chal$ which is randomly selected by the client and sent to the cloud, a file F and the set of authenticators Φ, and generates a proof P that the cloud has correctly preserved F.
- ProofVerify$(PK, Chal, P)\rightarrow$("True" or "False"): The proof verification algorithm is executed by the client to verify the proof generated by ProofGen$(PK, Chal, F, \Phi)$. It takes as input the public key PK, the same challenge $Chal$ used in ProofGen and the proof P, and outputs "True" or "False."

2.2 | Public key encryption

The public key encryption (PKE) scheme[23] consists of three polynomial-time algorithms (KeyGen, Enc, Dec) as follows:

- KeyGen(1^λ)$\rightarrow$ (pk, sk): The randomized key generation algorithm takes as input a security parameter λ and outputs the public key/private key pair (pk, sk).
- Enc$(pk, m)\rightarrow c$: The probabilistic encryption algorithm takes as input a message $m \in \mathcal{M}$ and a public key pk, then outputs a ciphertext c.
- Dec$(sk, c)\rightarrow m$: The deterministic decryption algorithm takes as input a ciphertext c and a private key sk, either outputs a message m or an error symbol $\perp$.

2.3 | Blockchain and smart contract

Blockchain is a new distributed infrastructure and computing paradigm based on transparent and trusted consensus rules.[24] It is composed of data blocks in the form of ordered chain in the peer-to-peer network environment. It uses the consensus algorithm to update data and uses cryptographic technologies to ensure tamper-resilience, unforgeability, and traceability of its data. Although researchers have proposed some methods[25-28] to attack blockchain, these attacks mainly focus on bitcoin. Blockchain is still considered as an effective technology to build trust in distributed nodes.[21,29,30]

In 1994, the concept of smart contract was first proposed by Szabo[31] which is defined as a kind of computer protocol that executes contract terms through information technology. Blockchain can supply a platform for supporting the automatical operation of programmable smart contracts by consensus nodes without any trusted third party once predefined rules have been met.[32]

2.4 | Overview of our solution

In the secure deduplication of encrypted data supporting client-side semantically secure encryption without the trusted third party, there are two types of the participants, that is, cloud storage server $\mathcal{S}$ and data owners $\mathcal{C}$. The data owners are divided into two categories as the original uploader and the subsequent uploader of the data. When an uploader $\mathcal{C}$ wants to upload to $\mathcal{S}$ a data file F which has been originally encrypted and uploaded to $\mathcal{S}$ by another uploader $\mathcal{C}_i$, there must be some mechanisms to ensure that $\mathcal{C}$ can obtain the encryption key K_F used by $\mathcal{C}_i$. Meanwhile, other users without F cannot obtain K_F. In traditional client-side deduplication, there may be an additional independent server to assist the encryption key transfer from $\mathcal{C}_i$ to $\mathcal{C}$ which is unrealistic in practice.

As blockchain is naturally a decentralized system maintained by all nodes in the network, the core idea of our solution of transferring the encrypted key from $\mathcal{C}_i$ to $\mathcal{C}$ is to replace the trusted third party with smart contracts. The smart contracts play the roles of an encryption key holder and a checker for the integrity of the data stored in $\mathcal{C}$, while $\mathcal{C}$ plays the role of an integrity prover to prove that he really holds the data completely.

As shown in Figure 1, the original uploader of the data file F encrypts the data using a semantically secure symmetric encryption scheme, generates and writes into a smart contract the set of authenticators Φ for the plaintext of F by AuthGen algorithm of a cloud data integrity auditing protocol **Audit**. When another data owner of the data file F wants to upload F to the cloud storage, the cloud storage server redirects him to the smart contracts so that a data integrity auditing protocol is implemented between the uploader and the smart contracts. The uploader proves to the smart contracts that he really has the complete data of F as a proof of ownership. Then, he can obtain the data encryption key used by the original data uploader from the smart contracts.

Specifically, to upload a data file F, the data owner first calculates a short hash value h of the file F and sends it to the cloud storage server. The cloud storage server uses the short hash value h to determine whether the data has already been stored in the cloud storage.

If the data file F has not been uploaded, the data owner is the original uploader of this file. He encrypts the data to get the ciphertext C of F by a semantically secure symmetric encryption scheme **SE** under a randomly selected key K. Then, he generates the set of authenticators Φ for F by AuthGen algorithm of a cloud data integrity auditing protocol **Audit**.

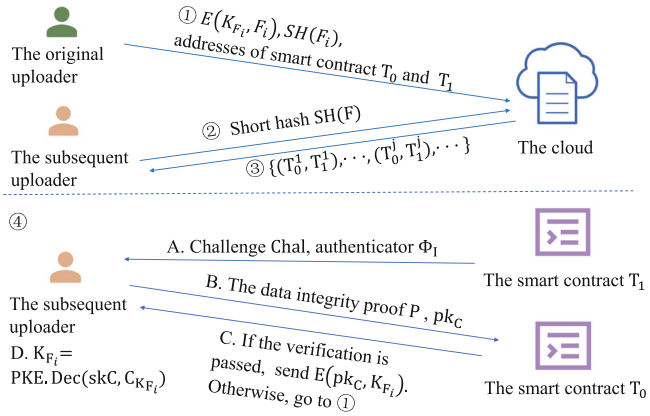


FIGURE 1 The uploading processes example

The ciphertext C of F is outsourced to the cloud storage server. The encryption key K and the code of ProofVerify algorithm of Audit are written into a data-privacy-preserving smart contract T_0 which can be programmed by some coding language for data-privacy-preserving smart contracts such as zkay.[33] The data owner also writes the set of authenticators Φ for F and his public key of **Audit** into a smart contract T_1. Then the data uploading process is done.

If the data file F has been uploaded, the data owner is a subsequent uploader of this file. Then, the cloud storage server redirects him to execute the smart contract T_1. Smart contract T_1 generates challenges *Chal* according to its current state and sends *Chal* and corresponding authenticators in Φ to the uploader. The uploader generates the data integrity proof P of F and executes the smart contract T_0 taking P as its inputs. T_0 implements data integrity verification by the ProofVerify algorithm of **Audit**. If the verification is passed, it means that the uploader actually preserves the data file F. Then, T_0 encrypts the data encryption key K using a PKE scheme under the public key of the uploader and sends the ciphertext C_K of K to the uploader. At last, the uploader can obtain the data encryption key K of the data file F by decrypting C_K under his own private key.

3 | THE PROPOSED PROTOCOL

In this section, we describe our blockchain-based deduplication protocol in detail. Let **PKE** be a public-key encryption scheme, and **Audit** be a cloud data integrity auditing protocol which is described in Section 2. Let *pFunc* be a pseudorandom function and *hFunc* be a hash function.

Step 1. File existence check. The idea of file existence check comes from Liu et al's work,[9] and we modify the data structure maintained by the storage server in their protocol[9] to make it applicable to our blockchain-based protocol. Specifically, when C wants to upload a data file $F = \{m_1, \ldots, m_n\}$ to S, it first sends a short cryptographic hash $sh = SH(F)$ of F to S so that it can check the existence of F in S storage. Note that due to the high collision rate of $SH()$, S cannot use sh to reliably guess the content of F offline.[9] Actually, since different files may have the same short hash, S can only check the existence of the ciphertext of the files $(F_1, F_2, \ldots, F_i, \ldots)$ whose plaintexts have the same short hash value due to the high collision rate of the hash function SH. S should record the index of files whose plaintext maps to the same short hash, the list of clients who have stored the same data file in the cloud, and the original uploader of a file. The data structure maintained by the storage server S is shown in Figure 2 which is similar to that in Liu et al's work.[9] For example, "C_1(Original)" in Figure 2 is the uploader who originally uploaded the ciphertext of the file F_1, which is the only encrypted copy of F_1 stored in the cloud storage. "T_0 address" and "T_1 address" in Figure 2 are the addresses of the two smart contracts programmed by "C_1(Original)." Our protocol should ensure that if $F = F_i$, the encryption key K_{F_i} can be securely transferred to C, where F_i is encrypted by another client C_j using a semantic-secure symmetric encryption scheme under the key K_{F_i}.

Step 2. According to the short hash value sh, S finds the original uploaders of the files in the set of $F_{sh} = \{F_1, F_2, \ldots, F_i, \ldots\}$ which sh is associated with. If $F_{sh} = \emptyset$, C is the original uploader of the data file F and goes to Step 4. Otherwise, it sends to C the set of all the addresses of the smart contracts programmed by these original uploaders denoted as $SC = \{(T_0^1, T_1^1), \ldots, (T_0^j, T_1^j), \ldots\}$, and goes to Step 3.

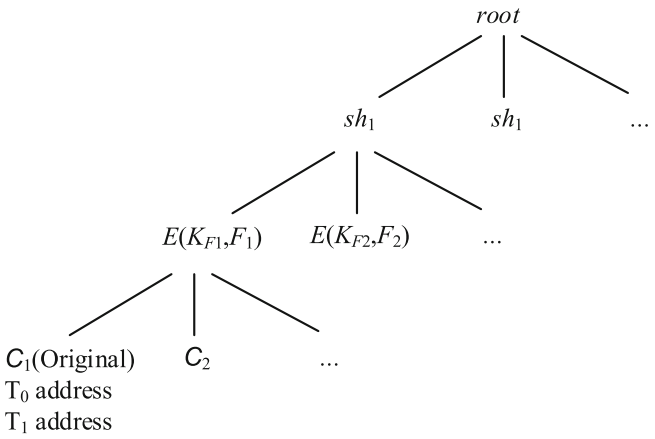


FIGURE 2 The record structure of server S

Step 3. For each $(T_0^j, T_1^j) \in SC$, C does the following operations:

C executes the smart contract T_1 deployed in T_1^j. Smart contract T_1 implements the following operations:

1. T_1 generates challenges *Chal* according to its current state τ as follows:
   - T_1 fixes an integer c.
   - T_1 constructs a set $I = \{s_1, \ldots, s_c\}$, where $s_i = pFunc(\tau||i)$, $i \in \{1, \ldots, c\}$.
   - T_1 generates the set of coefficients $\{v_1, \ldots, v_c\}$, where $v_i = hFunc(\tau||s_i)$, $s_i \in I$. Then, $Chal = \{(s_i, v_i)\}_{s_i \in I}$.
2. T_1 selects corresponding authenticators from Φ according to I. Let Φ_I be the set of corresponding authenticators.
3. T_1 outputs $Chal, \Phi_I$.

C generates data integrity proof P with the help of its own data file F by executing **Audit**.ProofGen$(PK, Chal, F, \Phi_I) \rightarrow (P)$. Let (pk_C, sk_C) be the public/private key pair generated by **PKE**.KeyGen$(1^\lambda) \rightarrow (pk_C, sk_C)$. C executes the smart contract T_0 deployed in T_0^j, and sends the data integrity proof P and pk_C to T_0 as its input.

Smart contract T_0 implements data integrity verification by executing **Audit**.ProofVerify$(PK, Chal, P)$. If the verification is passed which means that C actually preserves the data file F, T_0 generates the ciphertext $C_{K_{F_j}}$ of the encryption key K_{F_j} using **PKE**.Enc(pk_C, K_{F_j}), then sends $C_{K_{F_j}}$ to C.

C decrypts $C_{K_{F_j}}$ under its own private key to obtain K_{F_j}, that is, $K_{F_j} =$ **PKE**.Dec$(sk_C, C_{K_{F_j}})$. Go to Step 5.

If for all $(T_0^j, T_1^j) \in SC$, no integrity verification is passed, go to Step 4.

Step 4. In this step, it means that there doesn't exist a file satisfying that $F_i = F$ and $F_i \in F_{sh}$. C will be the original uploader of the data file F and does the following operations:

C executes **Audit**. SysSetup$(1^\lambda) \rightarrow (PK_C, SK_C)$.

C generates the set of authenticators Φ for F by executing **Audit**.AuthGen$(PK, SK, F) \rightarrow (\Phi)$.

C selects a random data encryption key K_F for F, then generates F's ciphertext $E(K_F, F)$.

C writes K_F and the code in Figure 3 into a smart contract T_0.

C writes Φ, PK_C and the code in Figure 4 into a smart contract T_1.

C uploads $E(K_F, F)$ and the addresses of smart contracts T_0 and T_1 to the cloud storage server S. Go to Step 5.

Step 5. The uploading process is finished.

Security proof. We will show that the execution of the deduplication protocol in the real model is computationally indistinguishable from the execution of F_{dedup} in the ideal model[9] as shown in Figure 5. We construct a simulator which can not only access F_{dedup} in the ideal model but also obtain messages that the corrupt parties would send in the real model. The simulator will generate a message transcript of the ideal model execution (IDEAL) which is computationally indistinguishable from that of the real model execution (REAL). In the proof, we assume that the smart contract T_0 and T_1 is implemented as an oracle to which the parties send their inputs and receive their outputs.

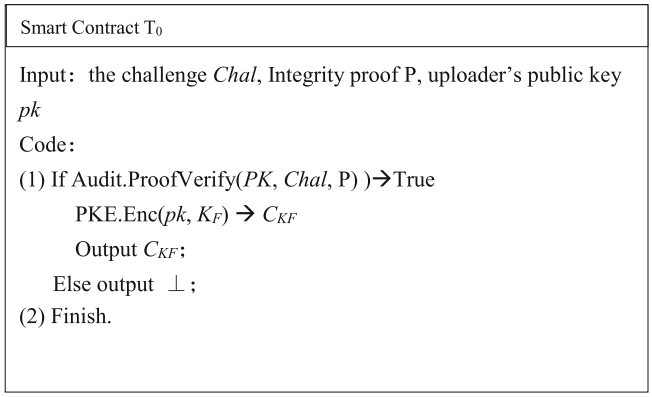


FIGURE 3 The smart contract T_0

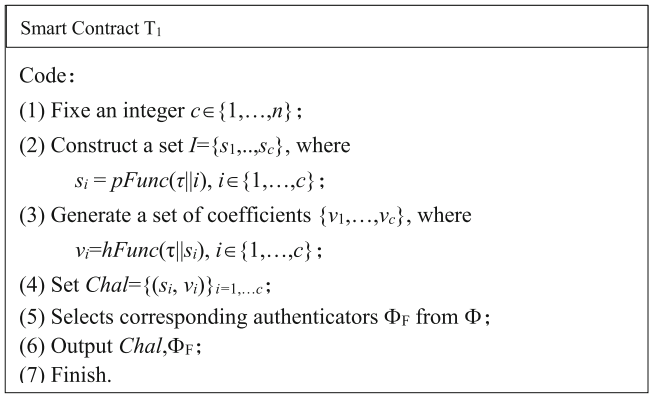


FIGURE 4 The smart contract T_1

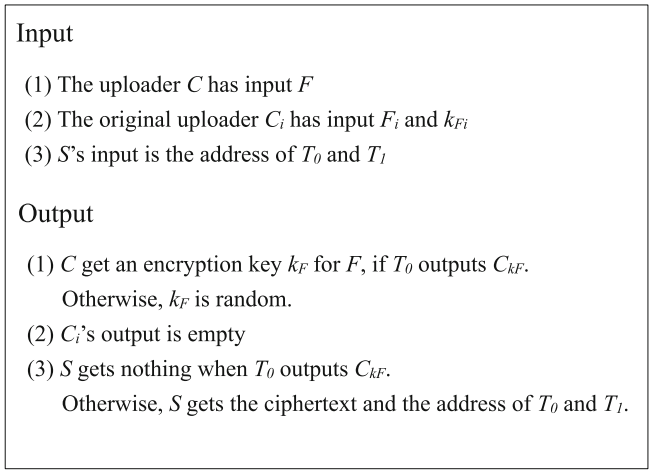


FIGURE 5 The ideal functionality F_{dedup} of deduplicating encrypted data

A corrupt subsequent uploader C: We first assume that the cloud storage server S and the original uploader C_i are honest and construct a simulator for uploader C. On receiving sh from C, the simulator observes the calls that C makes to the smart contracts T_0 and T_1. The simulator also records the output set $\{(P, F, sh)\}$. If C uses in that call the value P that appears in a set together with sh, the simulator invokes F_{dedup} with the file F appearing in that tuple. Otherwise, it invokes F_{dedup} with a random value. In either case, the simulator will obtain a key K_F(If F has been uploaded by any C_j, K_F is the key k_{F_j} for that file, otherwise K_F is a random value selected by C).

Suppose C is the subsequent uploader, that is, the file F already exists in the server. The simulator also records the output set $\left\{\Phi'_j\right\}$ that C receives from smart contract T_1. C executes **Audit**.ProofGen(PK, $Chal$, F, Φ) with the help of its own file F to get the proof $\{P_j\}$ and public key encryption **PKE**.KeyGen$\left(1^\lambda\right)$ to get (pk_C, sk_C). On receiving the P and pk_C from C, it chooses to check if the smart contract T_0^i's final result is C_{K_F}. If so, it calculates $e' = Enc\left(pk_c, C_{K_F}\right)$ and sends it back to C. Otherwise, it randomly selects an encryption key k, then sends it to the uploader C.

We now show that $IDEAL_C = \langle\left\{\Phi'_j\right\}, e'\rangle$ and $REAL_C = \langle\left\{\Phi'_j\right\}, e\rangle$ are identically distributed. (1) If the smart contract T_0's result is true and C behaves honestly, then $K_F = K_{F_j}$ and consequently e' and e are computationally indistinguishable. (2) If T_0's result is false, then K_F is a random value and the structure of e and e' are similar. (3) If C deviates from the protocol then the only action it can take, except for changing its input, is to replace some elements of $\{P_i\}$ that it sends to T_0. In the real model the execution will change, because the P_i is replaced by C. As a result, C will get a random value even though it inputs an existing file. In the ideal model, the same result will be caused by the event that a replaced element is chosen by the simulator. Based on (1) to (3), we can conclude that $IDEAL_c$ and $REAL_c$ are identically distributed.

A corrupt original uploader C_i: We prove security with relation to the relaxed functionality, where C_i also learns whether the uploaded file has the same short hash as F_i. The simulator needs to extract C_i's input:$\left(K_{F_i}, \Phi_i, address\right)$.

The simulator first observes whether sh matches the short hash of C_i. If not, it randomly chooses K_F and sends it to the C. C uses the key K_F to encrypt file F, getting ciphertext E. C writes K_F and **Audit**.ProofVerify(PK, $Chal$, P) to smart contract T_0, Φ and PK_C to T_1, $E\left(K_F, F\right)$ and the addresses of and T_0 and T_1 to the server S. The corrupt C_i may send the wrong Φ to smart contract T_1. As a result, when the subsequent uploader executes **Audit** to prove the integrity of his own F, the result of T_0 is still false. If so, the subsequent uploader will randomly choose the encryption key.

To show that the simulation is accurate, we observe that (1) if C_i behaves honestly, then $IDEAL_{C_i}$ and $REAL_{C_i}$ are obviously indistinguishable; (2) if C_i deviates from the protocol, the only operation it can do is to send wrong values to T_1, as a result, the proof P generated in the subsequent process is also wrong in both the real and ideal model. If P is wrong, T_0's final result must be false and K_F is assigned a random value. Based on (1) and (2), we can conclude that $IDEAL_{C_i}$ and $REAL_{C_i}$ are identically distributed.

A corrupt server S: The simulator first sends a short hash to the server S. Then S selects and sends to the uploader C a set of addresses. The set contains the addresses of the smart contracts T_0 and T_1 deployed by the original uploaders whose uploaded files are with the same short hash values. After executing the smart contracts, if the result of T_0 is true, the ciphertext C_{K_F} of encryption key K_F is sent to C, otherwise, an error is sent. The corrupt S may send to C a set of random values $SC = \left\{\left(T_0^1, T_1^1\right), \ldots, \left(T_0^j, T_1^j\right), \ldots\right\}$ and observes the outputs.

Next, the simulator invokes F_{dedup}. It detects the result of T_0, if it receives $E\left(pk_C, K_F\right)$ it sets $b = 1$; otherwise, it sets $b = 0$. If $b = 1$, it means T_0 passes **Audit**.ProofVerify(PK, $Chal$, P) and passes the result C_{K_F} to C, and the simulator receives $T_0's$ output C_{K_F}. If $b = 0$, the simulator receives $Enc(k_F, F)$, and the addresses of T_0 and T_1.

We now show that $IDEAL_S = \langle sh, \left\{\left(T_0^i, T_1^i\right), C_{K_{F_i}}, F\right\}\rangle$ and $REAL_S = \langle sh, \left\{\left(T_0^i, T_1^i\right), C'_{K_{F_i}}, E\left(K_F, F\right)\right\}\rangle$ are identically distributed. (1) Because $\left\{C_{K_{F_i}}, C'_{K_{F_i}}\right\}$ is generated by the smart contract T_0, $C_{K_{F_i}}$ cannot be distinguished from $C'_{K_{F_i}}$ in both ideal and real models. (2) If F exists and S behaves honestly, it will get the same $C_{K_{F_i}}$ in the real model and the ideal model. (3) If F does not exist in S, then K_F is the chosen random value in the real model, so S cannot distinguish $Enc(K_F, F)$ from a random string. Also, in the ideal model, the simulator gets from F_{dedup} the encryption of F under a random key. (4) Since S does not have the private key sk_C of an uploader C, it is impossible for S to decrypt the output C_{K_F} of T_0 in both of the ideal and the real models. Based on (1)-(4) we can conclude that $IDEAL_S$ and $REAL_S$ are identically distributed.

A collusion between a corrupt uploader and a corrupt server: The simulator acts to be C and S to call F_{dedup} inputting C's file F and public key PK_c. Here, S has no input to F_{dedup}. Then, the simulator receives the outputs of C and S. This situation is similar to the situation where the uploader is corrupted, except that S can choose a series of $\left\{\left(T_0^i, T_1^i\right)\right\}$ to implement the cloud data integrity audit protocol.

4 | RESULTS AND ANALYSIS

In this section, we analyze the performance of the protocol. The smart contracts on the Hyperledger Fabric are written in the Go language, and they are packaged, installed, and invoked in the test environment to implement our protocol. Figure 6 summarizes the basic information of our experiments.

4.1 | Experiment deployment

In this subsection, we describe how to implement the experiments. In our experiments, we use the auditing protocol proposed by Shacham and Waters[34] as it is a very simple and efficient protocol which is suitable to be used in the smart contract.

- First, we divide the file into N blocks, each of which is M kB in size.
- Randomly select a 256-bit prime number p to perform modulo operation with each block, and finally get an integer set $\{m_i\}_{i\in[1,N]}$ within 256 bits.
- Calculate file block corresponding authenticator $\{\Phi_i\}_{i\in[1,N]}$, where $\Phi_i = f_k(i) + \alpha m_i \in \mathbb{Z}_p$, $\alpha \in Z_p$. The function $f()$ is HMAC-SHA256, and k is its random key.
- Randomly select S blocks from $\{m_i\}_{i\in[1,N]}$. In the smart contract T_1, the corresponding $Chal(Q) = \{(s_i, v_i)\}_{i\in[1,S]}$ is calculated using the pseudorandom function *pFunc* and the hash function *hFunc*.
- The client generates the proof $P = (\Phi, \mu)$, where $\Phi \leftarrow \sum_{(s_i,v_i)\in Q} v_i \cdot \Phi_i$ and $\mu \leftarrow \sum_{(s_i,v_i)\in Q} v_i \cdot m_i$.
- Pass P and the random key k to the smart contract for verification, the verification formula is $\Phi \stackrel{?}{=} \alpha \cdot \mu + \sum_{(s_i,v_i)\in Q} v_i \cdot f_k(i)$. The final verification result is true or false.
- When the result is true, we encrypt the data encryption key K_F in the smart contract T_0 using RSA public-key encryption algorithm under the public key given by the uploader.

We implemented three experiments.

1. In the first experiment, we studied the duration of the authenticator and proof generation when files of different sizes were uploaded. We selected a total of 10 groups of file data from 10M to 100M and fixed the size of each file block to 100k, and the number of selected checked data blocks was 100. The results are shown in Figure 7.
2. In the second experiment, we evaluated the duration of authenticator and proof generation when dividing files into blocks of different sizes. Let the file size be 100M and the number of selected checked data blocks be 100. We set the block size from 100k to 1000k, that is, the file is divided into 1000 down to 100 blocks. The results are shown in Figure 8.
3. In addition, we tested the duration of challenge generation in the both of the above experiments.
4. In the last experiment, we evaluated the duration of proof verification of different numbers of checked data blocks in smart contracts, wherein we fixed the file size to 100M, the block size to 100k, and selected 100 to 1000 blocks.

CPU Series	Intel(R) Core(TM) i7-8700
RAM	16.0 GB (15.8 GB is available)
Operate System	Windows 10
	Ubuntu 18.04.2
Compiler Version	Microsoft Visual Studio 2019
	Fabric version 2.2.1
Environment	Go version: Go 1.14.4 linux/amd64
	Docker version 20.10.2
	Docker-compose version 1.17.1

FIGURE 6 Basic information

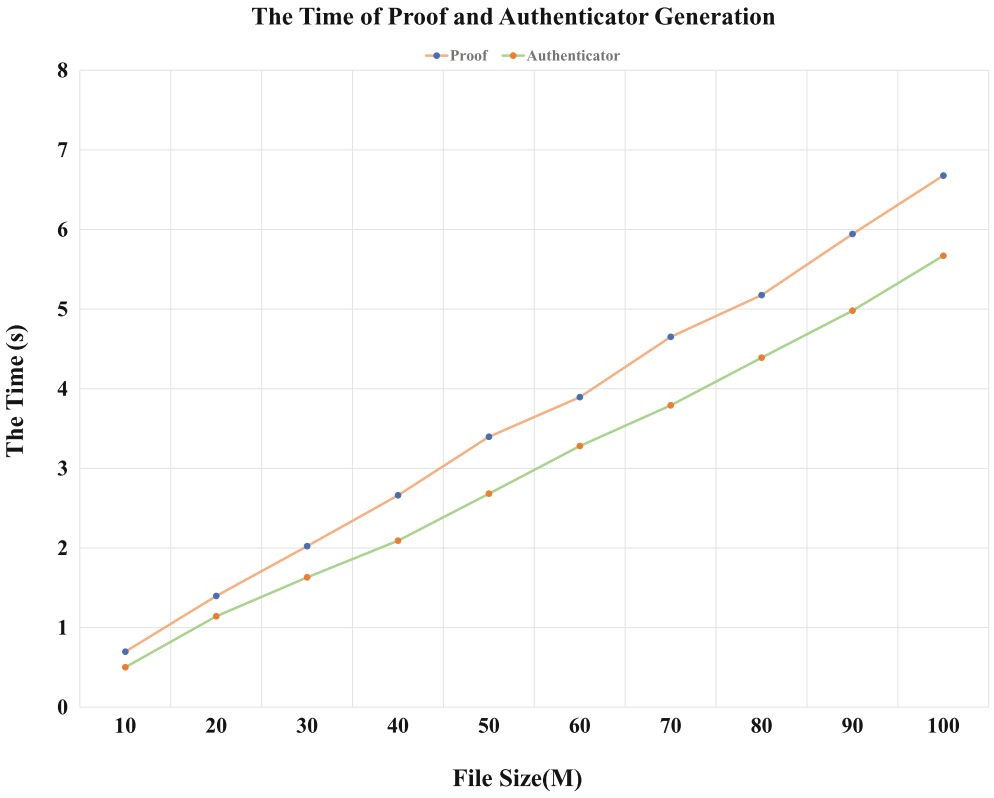


FIGURE 7 The time of proof and authenticator generation with different size of files

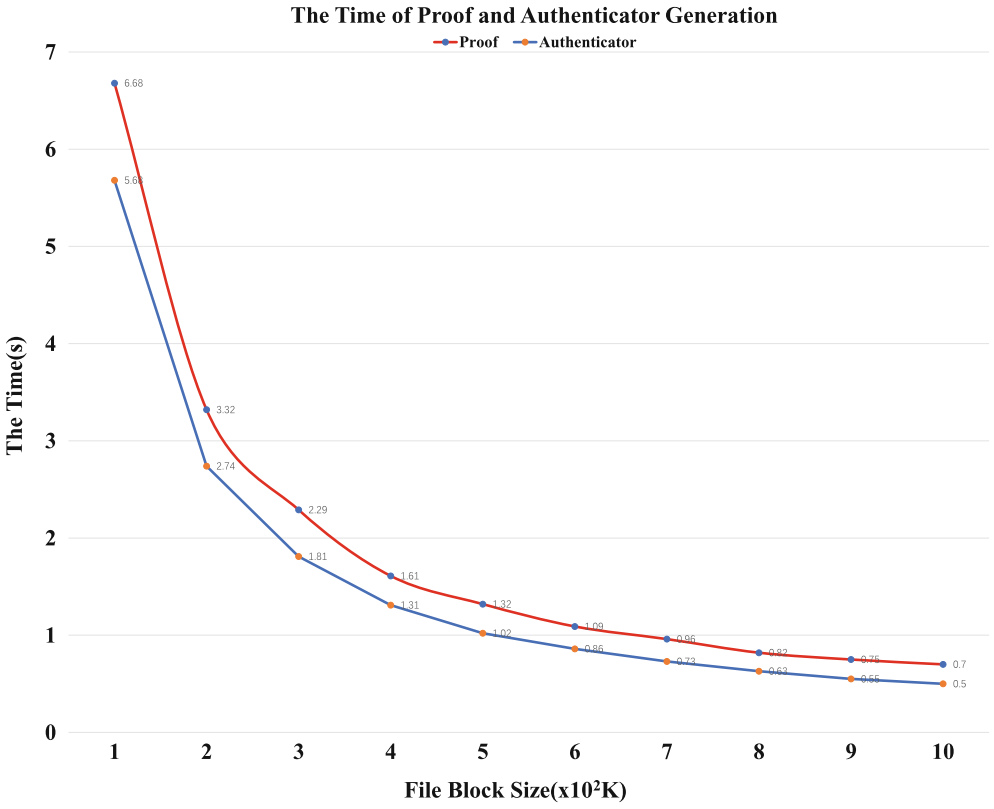


FIGURE 8 The time of proof and authenticator generation with different size of file blocks

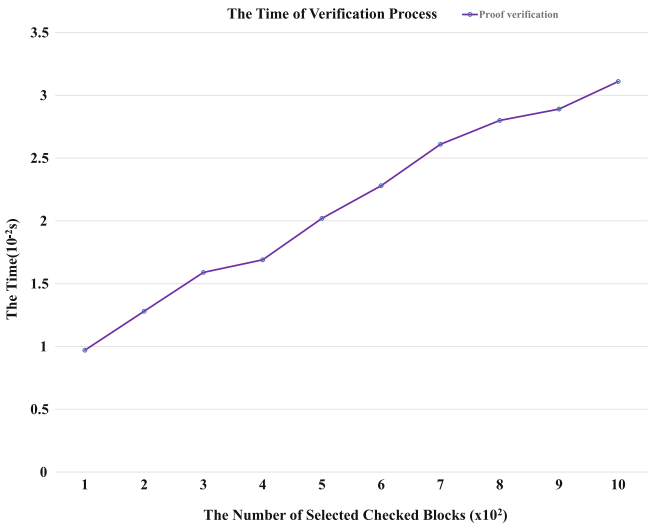


FIGURE 9 The time of verification process with different number of selected checked blocks

4.2 | Performance analysis

The duration of authenticator generation is the sum of the time to calculate corresponding authenticator for each block, and the duration of proof generation is the sum of the time to calculate Φ and μ.

From Figure 7, we can demonstrate that the durations of the authenticator and proof generation are linearly related to the size of the uploaded file.

The most interesting aspect of Figure 8 is that the duration of the authenticator and proof generation is negatively correlated with the block size. Through the analysis of the experimental data, it can be seen that the larger the file block and the fewer the number of blocks, the less time to generate the authenticator and the less time it eventually consumes. In addition, the fewer the number of blocks, the less time of the proof generation.

From the description of the proposed protocol, it can be obtained that the duration of the challenge generation is linear with the number of selected checked data blocks rather than the file size or the block size. By the implementation, we can get that an element $(s_i, v_i) \in Chal(Q)$ can be generated in the smart contract in about 15 milliseconds.

In the last experiment, we added a timestamp to the chain code to record the start time of the contract call and calculated the duration of the verification process according to the end time of the contract call and the time stamp. The experimental results are shown in Figure 9.

We also analyze the running time overhead of the PKE process in smart contract T_0 which takes about 26 milliseconds when using the RSA public-key encryption algorithm under a 1024-bit length key.

Finally, we make a comparison in terms of efficiency between our proposed protocol and the current deduplication protocols that support client-side semantically secure encryption and require the trusted third party or the previous uploader to be online. We focus on the scenario that the data to be uploaded is duplicated, and we compare the time required by each protocol when uploading a duplicated data of size 100 MB, that is, the time of obtaining the data encryption key. For the protocols requiring the trusted third party (key server),[12,13] the time of obtaining the data encryption key is independent of the data size. Using the benchmarks of the protocols[12,13] as references, to obtain the data encryption key, the protocol proposed in Bellare et al's work[12] needs about 82 milliseconds and the protocol proposed in Duan's work[13] needs about 500 milliseconds for the worst case due to the use of distributed key servers. For our solution, it relies on the original uploader and the smart contracts. The original uploader generates the authenticator when uploading the file, and the following uploader calculates the proof P and submits it to the smart contract for verification. According to the performance analysis, when the file size is 100 MB, the block size is 1000 KkB, and the number of selected check blocks is 100, the uploading process costs about extra 2.7 seconds besides the normal uploading process. Note that the original uploader needs to generate the authenticator Φ and deploy two smart contracts which costs about more than 0.5 seconds. Table 1 shows the differences between our protocol and the protocols in the compared works.[12,13] Therefore, we

TABLE 1 The comparison

Protocol	No third trusted party	Key request time (ms)
12	×	82
13	×	500
Our	✓	2700

can conclude that our protocol neither requires any trusted third party nor needs other uploaders to be online, and moreover it has reasonable running time overhead and high practicability, although we have to admit that our protocol is less efficient than the protocols require the trusted third party.[12,13]

5 | DISCUSSION

Privacy protection of the data encryption key K_F in the smart contract. Privacy protection has attracted more and more attention in various fields, such as machine learning,[35,36] data storage,[37] blockchain,[33] social networks,[38] etc. As is known to all, Hyperledger Fabric is a permissioned blockchain architecture, which provides a consistently distributed ledger, shared by a set of peers. The core principle of Hyperledger Fabric is that all the peers have the same view of the shared ledger, which makes it a challenge to support private data for the different peers.[39] It can be seen from Section 3 that the private data K_F is placed in the smart contract T_0. In this scenario, how can we ensure data privacy in the smart contract? There are many existing solutions.

Privacy protection can be achieved by specifying the privacy dataset. It is pointed out that for the implementation of the privacy database, if some organizations on the channel want to protect the data privacy and keep it secret to other organizations on the channel, a new channel can be created to allow only organizations with access rights to private data to join.[40] However, this method will incur additional management overhead. Zkay[33] is a system for specifying and executing data privacy in smart contracts. To implement zkay contract, it is proposed to automatically convert it into the contract that can be deployed on the public blockchain and is equivalent in terms of privacy and function. The implementation of the original prototype of zkay proves the feasibility of the method, but its proof of concept is seriously restricted, such as insecure encryption and lack of important language features. Later, the zkay v0.2[41] made up for the deficiency of zkay, which can support the most advanced asymmetric encryption and hybrid encryption. It introduces many new language features (such as function call, private control flow, and extension type support), allows different zk-SNARKs backends, and reduces compilation time and on-chain cost.

For Hyperledger Fabric, it can achieve data isolation by introducing multiple channels and private data collections. The multi-channel method separates the information between different channels. In theory, all nodes in the same channel share the data, but the data access permissions can be controlled by policies, which define the fabric system operation and data access permissions.[40]

From the above discussion, it can be seen that the existing blockchain privacy protection technologies can fully realize the privacy protection for the data encryption key K_F in the smart contract. The specific technologies will not be discussed in detail here.

6 | CONCLUSION

In this article, we mainly focus on how to use smart contracts to complete integrity auditing work, thereby realizing secure encrypted data deduplication. We deploy the proof verification process of the integrity auditing protocol in the smart contract and make the client generate the integrity proof of the data to be uploaded and send it to the smart contract for checking. Based on the characteristics of the blockchain, such as decentralization, non-tampering, traceability, multi-party maintenance, openness, and transparency,[42] it realizes the data ownership proof relying on the smart contract, which in turn enables encrypted data deduplication. Our protocol avoids the limitations of the existing encrypted data deduplication protocols which need a trusted third party or require that some previous uploaders be online to run the PAKE protocol. In addition, the run-time overhead of our protocol is reasonable, and it is much more practical.

It should be pointed out that the privacy-preserving smart contract plays an important role in our proposed protocol for the protection of data encryption key, which has not been implemented in our protocol. We leave it a future work.

ACKNOWLEDGMENTS

This research was supported by Shandong Provincial Natural Science Foundation (Nos. ZR2020LZH002, ZR2020MF055, ZR2021LZH007, ZR2020QF045), The Open Project of Key Laboratory of Network Assessment Technology, Institute of Information Engineering, Chinese Academy of Sciences (No. KFKT2019-002).

CONFLICT OF INTEREST

The authors declare no potential conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

1. Guarini D. Experts say Facebook leak of 6 million users'data might be bigger than we thought; 2013.
2. iCloud leaks of celebrity photos; 2014.
3. Recent major data leakage incidents at home and abroad; 2019.
4. Reinsel D, Gantz J, Rydning J. The digitization of the world from edge to core - data age 2025; 2018.
5. Jiang T, Chen X, Wu Q, Ma J, Susilo W, Lou W. Secure and efficient cloud data deduplication with randomized tag. *IEEE Trans Inf Forens Secur*. 2017;12(3):532-543.
6. Xu LJ, Hao R, Yu J, Vijayakumar P. Secure deduplication for big data with efficient dynamic ownership updates. *Comput Electr Eng*. 2021;96:107531. doi:10.1016/j.compeleceng.2021.107531
7. Douceur JR, Adya A, Bolosky WJ, Simon P, Theimer MM. Reclaiming space from duplicate files in a serverless distributed file system. ICDCS'02; 2002:617-624; IEEE, Vienna, Austria.
8. Bellare M, Keelveedhi S, Ristenpart T. Message-locked encryption and secure deduplication. *Advances in Cryptology – EUROCRYPT'15*. Athens, Greece: Springer; 2013:296-312.
9. Liu J, Asokan N, Pinkas B. Secure deduplication of encrypted data without additional independent servers. *CCS'15*. Denver, Colorado: ACM; 2015:874-885.
10. Abadi M, Boneh D, Mironov I, Raghunathan A, Segev G. Message-locked encryption for lock-dependent messages. *Advances in Cryptology – CRYPTO'13*. Santa Barbara, CA: Springer; 2013:374-391.
11. Bellare M, Keelveedhi S. Interactive message-locked encryption and secure deduplication. *PKC'15*. Gaithersburg, MD: Springer; 2015:516-538.
12. Bellare M, Keelveedhi S, Ristenpart T. DupLESS: server-aided encryption for deduplicated storage. *Sec'13*. Washington, DC: USENIX Association; 2013:179-194.
13. Duan Y. Distributed key generation for encrypted deduplication: achieving the strongest privacy. *CCSW'14*. Scottsdale, Arizona: ACM; 2014:57-68.
14. Puzio P, Molva R, Önen M, Loureiro S. ClouDedup: secure deduplication with encrypted data for cloud storage. *CloudCom'13*. Bristol, UK: IEEE; 2013:363-370.
15. Zhang S, Xian H, Wang Y, Liu H, Hou R. Secure encrypted data deduplication method based on offline key distribution. *J Softw*. 2018;29(7):1909-1921.
16. Zhang S, Xian H, Wang L, Liu H. Secure cloud encrypted data deduplication method. *J Softw*. 2019;30(12):3815-3828.
17. Xian H, Liu H, Zhang S, Hou R. Verifiable secure data deduplication method in cloud storage. *J Softw*. 2020;31(2):455-470.
18. Liu J, Duan L, Li Y, Asokan N. Secure deduplication of encrypted data: refined model and new constructions. In: 393, ed. *CT-RSA'18*. San Francisco, CA: Springer; 2018:374.
19. Chen H, Yu J, Zhou H, Zhou T, Liu F, Cai Z. SmartStore: a blockchain and clustering based intelligent edge storage system with fairness and resilience. *Int J Intell Syst*. 2021;36(9):5184-5209.
20. Sarkar A, Maitra T, Neogy S. *Blockchain in Healthcare System: Security Issues, Attacks and Challenges*. Cham: Springer International Publishing; 2021:113-133.
21. Guo J, Wang Y, An H, Liu M, Zhang Y, Li C. IIDQN: an incentive improved DQN algorithm in EBSN recommender system. *Sec Commun Netw*. 2022;2022:1-12. doi:10.1155/2021/7502248
22. Hu C, Xu Y, Liu P, Yu J, Guo S, Zhao M. Enabling cloud storage auditing with key-exposure resilience under continual key-leakage. *Inf Sci*. 2020;520:15-30. doi:10.1016/j.ins.2020.02.010
23. Naor M, Yung M. Public-key cryptosystems provably secure against chosen ciphertext attacks. *STOC'90*. Baltimore, Maryland: ACM; 1990:427-437.
24. Zhang D, Le J, Lei X, Xiang T, Liao X. Exploring the redaction mechanisms of mutable blockchains: a comprehensive survey. *Int J Intell Syst*. 2021;36(9):5051-5084.

25. Singh S, Hosen AS, Yoon B. Blockchain security attacks, challenges, and solutions for the future distributed IoT network. *IEEE Access*. 2021;9:13938-13959. doi:10.1109/ACCESS.2021.3051602
26. Wang Y, Wang Z, Zhao M, et al. BSMEther: bribery selfish mining in blockchain-based healthcare systems. *Inf Sci*. 2022;601:1-17. doi:10.1016/j.ins.2022.04.008
27. Li T, Wang Z, Chen Y, Li C, Jia Y, Yang Y. Is semi-selfish mining available without being detected? *Int J Intell Syst*. 2021:22. doi:10.1002/int.22656
28. Li T, Chen Y, Wang Y, et al. Rational protocols and attacks in blockchain system. *Sec Commun Netw*. 2020;2020(44):1-11.
29. Wang W, Xu H, Alazab M, Gadekallu TR, Han Z, Su C. Blockchain-based reliable and efficient certificateless signature for IIoT devices. *IEEE Trans Ind Inform*. 2022;18(10):7059-7067. doi:10.1109/TII.2021.3084753
30. Deebak BD, Memon FH, Khowaja SA, et al. Lightweight blockchain based remote mutual authentication for AI-empowered IoT sustainable computing systems. *IEEE Internet Things J*. 2022;1:9. doi:10.1109/JIOT.2022.3152546
31. Szabo N. Formalizing and securing relationships on public networks. *First Monday*. 1997;2(9):1-27.
32. Buterin V. A next generation smart contract and decentralized application platform. Ethereum white paper; 2014.
33. Steffen S, Bichsel B, Gersbach M, Melchior N, Tsankov P, Vechev M. Zkay: specifying and enforcing data privacy in smart contracts. *CCS'19*. New York, NY: Association for Computing Machinery; 2019:759-1776.
34. Shacham H, Waters B. Compact proofs of retrievability. *J Cryptol*. 2013;26:442-483.
35. Zhu T, Zhou W, Ye D, Cheng Z, Li J. Resource allocation in IoT edge computing via concurrent federated reinforcement learning. *IEEE Internet Things J*. 2022;9(2):1414-1426. doi:10.1109/JIOT.2021.3086910
36. Shokri R, Shmatikov V. Privacy-preserving deep learning. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security; 2015:1310-1321.
37. Gao X, Yu J, Chang Y, Wang H, Fan J. Checking only when it is necessary: enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. *IEEE Trans Depend Sec Comput*. 2022;19(6):3774-3789. doi:10.1109/TDSC.2021.3106780
38. Li J, Hu X, Xiong P, Zhou W. The dynamic privacy-preserving mechanisms for online dynamic social networks. *IEEE Trans Knowl Data Eng*. 2022;34(6):2962-2974. doi:10.1109/TKDE.2020.3015835
39. Benhamouda F, Halevi S, Halevi T. Supporting private data on hyperledger fabric with secure multiparty computation. *IBM J Res Develop*. 2019;63(2/3):3:1-3:8. doi:10.1147/JRD.2019.2913621
40. Ma C, Kong X, Lan Q, Zhou Z. The privacy protection mechanism of hyperledger fabric and its application in supply chain finance. *Cybersecurity*. 2019;2(1):1-9. doi:10.1186/s42400-019-0022-2
41. Baumann N, Steffen S, Bichsel B, Tsankov P, Vechev M. zkay v0.2: practical data privacy for smart contracts. Technical report; 2020.
42. Guo S, Wang R, Zhang F. Overview of the principles and applications of blockchain technology. *Comput Sci*. 2021;48:271-281.

How to cite this article: Qin G, Li L, Liu P, Hu C, Guo S. Blockchain-based secure deduplication of encrypted data supporting client-side semantically secure encryption without trusted third party. *Trans Emerging Tel Tech*. 2024;35(4):e4712. doi: 10.1002/ett.4712

Controlled Search: Building Inverted-Index PEKS With Less Leakage in Multiuser Setting

Guiyun Qin, Pengtao Liu, Chengyu Hu, Zengpeng Li, and Shanqing Guo

***Abstract*—The public key encryption with keyword search (PEKS) schemes are mostly applied to small data sets in mail forwarding systems. When retrieving large databases, the typical search mechanism makes them inefficient and impractical. When designing a PEKS scheme, except for remedying the vulnerability of keyword guessing attacks (KGAs), other leakage issues, such as multipattern privacy and forward/backward security are rarely considered, which may lead to information leakage. Moreover, most existing PEKS only consider applications in single-user scenarios, and cannot be directly transferred to multiuser scenarios, which undermines the value of data utilization. To cope with the above concerns, we propose a PEKS scheme based on an inverted index where the bitmap is used to build the index for the first time in PEKS to meet some seemingly conflicting yet desirable characteristics. First, it has high search efficiency under multiwriter and multiuser. Through linear transformation, users quickly retrieve data and control other users' access to their data without relying on a third party for authentication. Second, we prove its security in an enhanced security model that achieves multipattern privacy and forward and backward security. It can also resist KGA attacks without a designated tester, which makes it more practical. Finally, it can be extended to achieve search result verification. Compare to the scheme (Zhang et al. ICWS 2016), it has absolute advantages in security and computational cost where the search efficiency is improved by two orders of magnitude.**

***Index Terms*—Forward and backward security, inverted index, multisearchable management, public key encryption with keyword search (PEKS), result verification.**

I. Introduction

PUBLIC key encryption with keyword search (PEKS) [1], as an important branch of searchable encryption (SE) [2], focuses on solving the complex key management problems in symmetric SE (SSE). However, PEKS schemes currently suffer from the following deficiencies.

Due to the low efficiency of the typical PEKS schemes, they do not support the data owner sharing a large amount of data with the recipient, which limits the application scope of PEKS. The reason is that they always implement searching a keyword by testing all the keyword ciphertexts of all the files, i.e., the search efficiency depends on the total number of the keyword ciphertexts in all the files.

PEKS faces some security challenges and leakage problems emerge endlessly. Except for keyword guessing attacks (KGAs), the privacy of patterns and forward security are rarely considered which makes the adversary infer some sensitive information from trapdoors and search results (e.g., Leakage-Abuse Attack [3]). As far as we know, the trapdoor generation algorithms of many PEKS schemes (e.g., [4] and [5]) are deterministic, which reveals the search pattern. When searching for the same keywords, the server (tester) returns the same search results, which leads to the leakage of the access pattern. The size pattern privacy has not been considered in existing PEKS, which reveals the number of query results and the number of keywords contained in the update file. Forward security is mainly considered in SSE schemes to resist the file injection attack [6], but we show that this attack is easier to deploy in PEKS schemes. A common approach to resist KGA attacks is to require an additional designated tester, but it is not practical.

In addition, transferring PEKS to multiuser scenarios is conducive to improving data utilization. A straightforward approach is distributing the search key to other users. However, different recipients may have different search permissions (e.g., supervisors and employees, and doctors and patients). The naive approach cannot meet the above requirement and is prone to data abuse.

To sum up, a challenging and practical question we raise is: *How to break these limitations, improve efficiency and security and increase the usability of the PEKS systems?*

A. Motivations and Methodologies

To address the above issues in PEKS schemes, such as inefficiency, vulnerability to KGA attacks, patterns leakage, and the limitations of multiuser search, we construct an inverted-index-based PEKS scheme.

The data sender considers the "keyword-bitmap matrix" as the index structure and encrypts it through an integer vector homomorphic encryption (HE) algorithm, which facilitates the search/update and patterns hiding. Specifically: The elements in the index matrix represent the relationship between

Manuscript received 13 December 2022; revised 15 May 2023; accepted 1 June 2023. Date of publication 19 June 2023; date of current version 25 December 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFA1000600, and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2022LZH013, Grant ZR2020LZH002, Grant ZR2020MF055, Grant ZR2021LZH007, and Grant ZR2020QF045. *(Corresponding authors: Chengyu Hu; Zengpeng Li.)*

Guiyun Qin is with the School of Cyber Science and Technology, Shandong University, Qingdao 266237, China (e-mail: qin1997@mail.sdu.edu.cn).

Pengtao Liu is with the School of Cyberspace Security, Shandong University of Political Science and Law, Jinan 250013, China (e-mail: ptwave@163.com).

Chengyu Hu, Zengpeng Li, and Shanqing Guo are with the School of Cyber Science and Technology, Shandong University, Qingdao 266237, China, also with the Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China, and also with Quan Cheng Laboratory, Jinan 250103, China (e-mail: hcy@sdu.edu.cn; zengpeng@email.sdu.edu.cn; guoshanqing@sdu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2023.3287353

keywords and files, where the element in the ith row and the jth column represent whether the jth file contains the ith keyword. If the file contains a keyword, the element in the corresponding position is set to 1, otherwise, it is set to 0. When the recipient wants to search a keyword w_i, a straightforward strategy is constructing a $1 \times m$ query matrix $\mathbf{B}$ where the element at the corresponding position of the keyword w_i is set to 1 and other elements are set to 0. Then, in the ciphertext state, query matrix $\mathbf{B}$ and index matrix are tested to obtain the bitmap corresponding to the keyword w_i. Since each keyword corresponds to the same number of bitmaps in the index matrix (i.e., the returned ciphertext results are the same), size pattern hiding is achieved. In fact, the search process is a linear transformation operation process. Due to the randomness of search tokens and ciphertext results, search and access patterns hiding are achieved. As for the update, the update of the encrypted index is achieved by utilizing a homomorphic addition operation, and during this process, forward and backward privacy is protected. See Section III for details. In addition, since there are no specific keyword ciphertexts, it can also resist KGA attacks without a designated tester.

Furthermore, through the linear transformation operation, it can establish an effective isolation mechanism among multiple users to prevent the interference of malicious users. Specifically, in multiuser scenarios, the sender (or recipient) can set different data access authorities (keywords or files) for users to achieve multisearchable management. More concretely, according to different retrieval needs, the sender (or recipient) constructs different control matrices $\mathbf{G}$. Using the control matrices $\mathbf{G}$ and the original index to perform calculation operations in the ciphertext state to obtain new index ciphertexts for different users. When specifying access authority to files, the sender (or recipient) also can further revoke the files they do not want to share to achieve file management.

Although it seems that the integer vector HE (IVHE) scheme can be trivially applied for constructing search schemes in the cloud, existing schemes only briefly describe how to construct a forward-index-based search scheme [7], [8]. To the best of our knowledge, there is no inverted-index-based PEKS scheme constructed based on the IVHE.

B. Our Contributions

In this article, we make affirmative progress in the following areas. The contributions are summarized as follows.

1) *Distinctive Index Construction:* Unlike traditional index construction, we use the bitmap structure to build the inverted index. The bitmap structure has the advantages of minimum search complexity, easy updating, etc., which is usually used in the SSE schemes. We apply it to PEKS for the first time and make up for its shortcoming of requiring to limit the number of files. Because the scheme is not necessary to test each keyword ciphertext of files, the keyword search is much more efficient. It also has "Noninteractive" search/update operations. No additional roundtrip between any parties is needed, except one inevitable roundtrip for the data owner to authorize the server to update the index, so it leaks less information.
2) *Multisearchable Management:* We realized a reliable isolation mechanism to ensure that individual users do not interfere with each other. In other words, it controls the retrieval authorities of the recipients neither requiring a trusted third party (or the data owner acting as a trusted third party) nor predetermining the number of users.
3) *Enhanced Security:* We present an enhanced security model—*Semantic Security*. It implies search/access/size pattern privacy, forward/backward privacy,[1,2] and resists KGA without a designed tester. We also show how to extend our scheme to verify the integrity of the search result.

C. Related Work

PEKS: Boneh et al. [1] proposed the first PEKS scheme via identity-based encryption (IBE) [9] for different data owners to send a small amount of data to a single user. Abdalla et al. [10] showed how to transform an anonymous IBE scheme into a PEKS scheme. Subsequently, various PEKS schemes are proposed.

Pattern Privacy: However, most PEKS schemes do not consider *search pattern* privacy and *access pattern* privacy, which leaks relevant information about queried keywords. The works [11], [12] claim that search pattern privacy can be preserved by introducing a security definition called *trapdoor indistinguishability* which makes the trapdoor generation algorithm probabilistic. It ensures that even if adversaries obtain a trapdoor, they cannot obtain the keyword information contained in the trapdoor. That is, there is a negligible advantage for any polynomial-time adversary A to correctly guess the trapdoor of keywords w_0 and w_1. Yet, search results for the same keyword are the same, so the adversary can still learn the search pattern by the access pattern. In addition, there is also *ciphertext indistinguishability* which prevents adversaries from obtaining keyword information contained in encrypted files without learning the keyword trapdoor. That is, there is a negligible advantage for any polynomial-time adversary A to correctly guess the ciphertext of keywords w_0 and w_1. Even if the *trapdoor and ciphertext indistinguishability* are achieved, the adversary can still distinguish the search results of different keywords by size pattern which is mainly considered in SSE.

KGA Attacks: Likewise, the typical PEKS schemes are vulnerable to KGAs [13] which can also be regarded as attack methods to get search pattern information in our opinion. To defend against KGAs, Rhee et al. [11] proposed a PEKS scheme with a designated tester. However, Wang et al. [14]

[1]Forward privacy ensures that the newly added file containing the keyword w cannot be linked to the trapdoor generated for the same keywords w in previous search operations. It is not considered in existing PEKS schemes, but we point out in Section III-B that it can be utilized by the server to get some private information about the outsourced data.

[2]Backward privacy ensures that searches for the keyword w cannot be linked to the files containing the keyword w which have been deleted. Although backward privacy needs not to be considered in traditional PEKS schemes, it is an important issue in the inverted-index-based schemes.

subsequently pointed out that it could not resist KGA attacks from internal malicious servers. Huang and Li [4] introduced an authenticated PEKS scheme that can resist internal KGA attacks. However, its trapdoor generation algorithm is deterministic, which reveals the search pattern. Qin et al. [5] presented the definition of multiciphertext indistinguishability (MCI), and improved Huang's scheme [4], while the improved scheme does not satisfy search pattern privacy either. Pan and Li [15] proposed a PEKS scheme that achieves *Multiciphertext/Trapdoor Indistinguishability*, but it also cannot hide the search pattern from the internal malicious server.

Efficiency: To improve the efficiency, some work attempts to build PEKS schemes using the inverted index with sublinear search efficiency which is commonly used in SSE schemes. Wang et al. [16] claimed to propose a PEKS scheme based on an inverted index. However, Wang et al. [17] proved that the scheme [16] cannot perform the keyword search correctly and modified it to be a new SSE scheme. Zhang et al. [12] proposed the first PEKS scheme based on the inverted index. Although the search efficiency has been greatly improved, its trapdoor generation efficiency is inefficient. In terms of security, although it claims to hide the search pattern, the search results reveal the access pattern. In addition, the file update method is interactive, which requires multiple rounds of communication between the user and the server.

Multiuser: To improve data utilization, multiuser PEKS schemes are proposed. The trivial method is that the owner creates the encrypted index for each receiver, which causes huge computational and communication overheads. There are currently two main paths [18]. One is relying on a third party (or the data owner to act as a third party (e.g., [19]) to generate search trapdoors for users (e.g., [20], [21], and [22]). The other path requires predetermine the number of users and precompute some parameters by all the users (e.g., [23] and [24]). The above two paths either have some leakage (eavesdropping and replay attacks) or are not flexible enough (requiring real-time online, etc.). How to break through them is still pending.

According to what we are informed, no PEKS schemes can achieve quick search efficiency while maintaining high security and can be extended to multiuser settings.

HE allows a third party (e.g., the cloud) to perform certain computational operations on encrypted data while retaining the functional and format characteristics of the encrypted data [25]. Rivest et al. [26] first proposed *homomorphism* as a possible solution to the problem of computing encrypted data without decryption. Subsequently, researchers have developed Partially HE algorithms that satisfy multiplication or addition (e.g., [27], [28], and [29]). Until, Gentry [30] proposed the first available fully HE (FHE) scheme, which supports both addition and multiplication operations. Brakerski et al. [31] proposed a parameterizable somewhat FHE scheme (BGV), whose security is based on LWE. It utilizes the key-switching technology to control the explosive growth of the ciphertext dimension during multiplication by restoring the expanded ciphertext dimension to the original dimension. At the same time, the modular-switching technology is used to replace the "Bootstrapping" process in [30] to control noise growth. Later, Brakerski [32] proposed the BFV scheme which only needs to solve the ciphertext dimension expansion problem through the key-switching technology. The computational cost and communication overhead of the above-mentioned technologies are still relatively high. To reduce the overhead, Peikert et al. [33] proposed a ciphertext packing technique (PVW). The cost of encrypting n-bit data in this scheme is basically the same as the cost of encrypting one-bit data in the Regev cryptosystem [34]. However, this scheme still requires decomposing the computational task into binary operations, which is expensive. In 2015, inspired by the above research, Zhou and Wornell [7] proposed an IVHE scheme. Unlike other general schemes, this scheme focuses on solving HE of a specific data type (integer vectors). Using the key-switching matrix to change the key-ciphertext pair, this scheme realizes complex operations in the encryption domain (e.g., linear transformation). Since there is no need to decompose the computational tasks into binary operations in the calculation process, it achieves better performance than FHE schemes, which is the main reason why we choose this algorithm.

Roadmap: In Section II, we introduce the preliminaries. In Section III, we present the framework and security definition of the scheme. In Section IV, we show the construction details and give how to implement multisearchable management. In Section V, we evaluate the experimental performance and make comparisons with typical schemes. Section VI discusses how to extend our solution to verify the integrity of the search result. Finally, we draw a conclusion in Section VII.

II. Preliminaries

A. Notation and Operation

$\mathrm{ID} = (id_1, id_2, \ldots, id_c)$ represents the file identifiers of c files, and $W = (w_1, w_2, \ldots, w_z)$ represents the keyword space of size z. $id_j = \{w_{j_1}, w_{j_2}, \ldots, w_{j_v}\}$ represents the file id_j and the keywords contained in it. In other words, each id_j is associated with a keyword list $W_j \subseteq W$, which contains all keywords in the file id_j. Below, we give the related definitions that will be used in the following parts.

1) For $a \in \mathbb{R}$, define $\lceil a \rfloor$ to round a to the nearest integer.
2) For vector $\mathbf{a} \in \mathbb{R}^n$, define $\lceil \mathbf{a} \rfloor$ to round each entry a_i of $\mathbf{a}$ to the nearest integer, and $|\mathbf{a}|$ represents the magnitude of vector $\mathbf{a}$: $|\mathbf{a}| = \max_i\{|a_i|\}$.
3) For matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, $|\mathbf{A}|$ represents the magnitude of matrix $\mathbf{A}$: $|\mathbf{A}| = \max_i\{|A_{ij}|\}$.
4) For $\mathbf{v} \in \mathbb{Z}_q^m$ (v_i is an entry of $\mathbf{v}$), $\mathbf{t}(t(\mathbf{v}))$ represents the "center" vector for $\mathbf{v}$, where $t(v_i) = \lfloor v_i \cdot q/p \rceil \in \mathbb{Z}_q$ is ith entry of $t(\mathbf{v})$.

Key-Switching Operation: According to [32], we introduce two decompositions.

1) *BitDecomp($\mathbf{y}$):* For $\mathbf{y} \in \mathbb{Z}_q^n$, let $\mathbf{x}_i \in \{0,1\}^n$ be such that $\mathbf{y} = \sum_{i=0}^{\lceil \log q\rceil -1} 2^i \cdot \mathbf{x}_i (\mathrm{mod} q)$, and output the vector $(\mathbf{x}_0, \ldots, \mathbf{x}_{\lceil \log q\rceil -1}) \in \{0,1\}^{n\lceil \log q\rceil}$.
2) *PowersOfTwo($\mathbf{u}$):* For $\mathbf{u} \in \mathbb{Z}_q^n$, output $[(\mathbf{u}, 2 \cdot \mathbf{u}, \ldots, 2^{\lceil \log q\rceil -1} \cdot \mathbf{u})]_q \in \mathbb{Z}_q^{n\cdot\lceil \log q\rceil}$.

Therefore, we can obtain

$$\langle \mathbf{y}, \mathbf{u}\rangle = \langle\ \mathrm{BitDecomp}\ (\mathbf{y}),\ \mathrm{PowersOfTwo}\ (\mathbf{u})\rangle\ (\mathrm{mod} q).$$

When the keys are all vectors, the key-switching matrix [32], [35] can switch the key-ciphertext pair to another key-ciphertext pair. To make it more practical, the work [7] gives a matrix version of key-switching composition to realize complex operations (e.g., linear transformation) of IVHE. For concrete, there is the following relation for the key $\mathbf{S} \in \mathbb{Z}_q^{f\times g}$ and the ciphertext $\mathbf{c} \in \mathbb{Z}_q^g$:

$$\mathbf{S}^*\mathbf{c}^* = \mathbf{S}\mathbf{c}.$$

Specifically, the intermediate ciphertext $\mathbf{c}^*$ means that each element c_j in $\mathbf{c}$ is represented as $c_j = \sum_{i=0}^{\ell-1} 2^i \cdot x_{j_i}$ according to *BitDecomp*. Therefore, $\mathbf{c}^* = [x_{1_0}, x_{1_1}, \ldots, x_{g_{\ell-2}}, x_{g_{\ell-1}}]^T \in \{0,1\}^{g\ell}$. According to *PowersOfTwo*, the intermediate key $\mathbf{S}^* \in \mathbb{Z}^{f\times g\ell}$ means that each element S_{ij} in $\mathbf{S}$ is replaces with $S_{ij} = [S_{ij}, 2\cdot S_{ij}, \ldots, 2^{\ell-1}\cdot S_{ij}]$.

Nextly, switch the original key $\mathbf{S} \in \mathbb{Z}_q^{f\times g}$ into the "target" key $\mathbf{S}' = [\mathbf{I}, \mathbf{T}] \in \mathbb{Z}^{f\times g'}$ where $\mathbf{I}$ is the identity matrix. To do this, a key-switching matrix $\mathbf{M} \in \mathbb{Z}^{g'\times g\ell}$ is constructed. It satisfies

$$\mathbf{M} = \begin{pmatrix} \mathbf{S}^* + \mathbf{E} - \mathbf{T}\mathbf{A} \\ \mathbf{A} \end{pmatrix} \bmod q$$

where $\mathbf{T}$ and $\mathbf{A} \in \mathbb{Z}^{(g'-f)\times g\ell}$ are the random matrices, and $\mathbf{E}$ is a noise matrix with small magnitude.

Finally, the ciphertext $\mathbf{c}$ is switched into the final ciphertext $\mathbf{c}'$ corresponding to the targe key $\mathbf{S}'$ which is defined as follows:

$$\mathbf{c}' = \mathbf{M}\mathbf{c}^* \bmod q \in \mathbb{Z}_q^{g'}.$$

B. Cryptographic Building Blocks

PEKS: There are four polynomial-time algorithms in a typical PEKS scheme.

1) *KeyGen*(λ) $\rightarrow$ (pk, sk): Input a security parameter λ and output the public-private key pair (pk, sk).
2) *PEKS*(pk, w) $\rightarrow$ C_w: Input a keyword w and the public key pk, and output the keyword ciphertext C_w.
3) *Trapdoor*(sk, w_i) $\rightarrow$ T_{w_i}: Input a keyword w_i and the private key sk, and output the trapdoor T_{w_i}.
4) *Test*(C_w, T_{w_i}) $\rightarrow$ c_i: Input the keyword ciphertext C_w and the trapdoor T_{w_i}, and output the result set c_i.

The data sender encrypts each keyword in the files using the public key of the recipient and sends the ciphertexts to the cloud server. Then, the recipient generates a search trapdoor based on its private key and the keyword to be searched. After receiving the trapdoor, the server executes the *Test* algorithm to match the keyword ciphertexts with the trapdoor.

IVHE: As an extension of the PVW scheme [33] from binary vectors to integer vectors, it is a public key cryptosystem.

Compared with the traditional HE schemes, IVHE is easier to implement complex operations, e.g., linear transformation. Their working mechanism is shown in Fig. 1.

The IVHE scheme consists of three polynomial time algorithms for encrypting the index matrix.

Key Generation: The secret key sk is $\mathbf{S} = [\mathbf{I}, \mathbf{T}] \in \mathbb{Z}_q^{m\times n}$, and the public key pk is $(\mathbf{A}, \mathbf{Q})$. $\mathbf{A} \in \mathbb{Z}_q^{(n-m)\times l}$ is a random matrix and $\mathbf{Q} = \mathbf{T}\mathbf{A} + \mathbf{E}$, where the elements in $\mathbf{E} \in \mathbb{Z}_q^{m\times l}$ come from the error distribution.

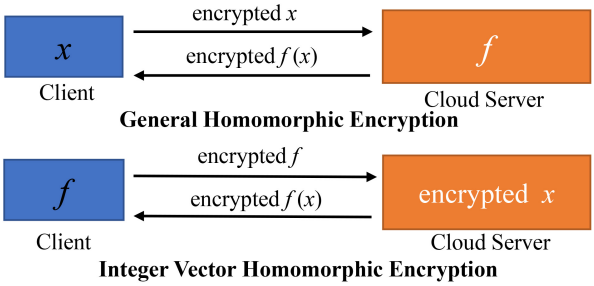


Fig. 1. Function and data.

Encryption: After encrypting the integer vector $\mathbf{v} \in \mathbb{Z}_p^m$, we get ciphertext $\mathbf{c} = (\mathbf{A}\mathbf{e}, \mathbf{Q}\mathbf{e} + \mathbf{t}(t(\mathbf{v}))) \in \mathbb{Z}_q^n$, where $q \gg p$ and $\mathbf{e} \leftarrow \{0,1\}^l$ is the noise vector.

Decryption: From [32], the vector $\mathbf{v} \in \mathbb{Z}_p^m$, the ciphertext $\mathbf{c}$, and the secret key $\mathbf{S}$ satisfy the equation

$$\mathbf{S}\mathbf{c} = w\mathbf{v} + \mathbf{e} \ (mod\ q) \tag{1}$$

where w is a large integer, that is, $w > 2|\mathbf{e}|$, $\mathbf{e}$ indicates that the noise vector belongs to the error term.

Therefore, the decryption process is represented as follows:

$$\mathbf{v} = \left\lceil \frac{\mathbf{S}\mathbf{c}}{w} \right\rfloor_q. \tag{2}$$

Addition Operation: If the ciphertexts $\mathbf{c_1}$ and $\mathbf{c_2}$ have the same key, $\mathbf{c_1} + \mathbf{c_2} = Enc(\mathbf{x_1} + \mathbf{x_2})$. When adding or deleting a file, we utilize the addition operation to achieve it.

Linear Transformation: In our solution, we achieve the keyword search and multisearchable management by the linear transformation operation $\mathbf{G}\mathbf{v}$. According to (1), for any matrix $\mathbf{G} \in Z^{m'\times m}$, it satisfies

$$(\mathbf{G}\mathbf{S})\mathbf{c} = w\mathbf{G}\mathbf{v} + \mathbf{G}\mathbf{e} \bmod q$$

$\mathbf{c}$ is regarded as the ciphertext of the plaintext $\mathbf{G}\mathbf{v}$ under the key $\mathbf{G}\mathbf{S}$. The client uses a key-switching matrix $\mathbf{M}$ to switch the key $\mathbf{G}\mathbf{S}$ into the new key $\mathbf{S}' = [\mathbf{I}', \mathbf{T}']$. After receiving $\mathbf{M}$, the server computes the new ciphertext $\mathbf{c}'$ under the new key $\mathbf{S}'$.

III. Framework and Security Definition

A. System Framework

To construct an inverted-index-based PEKS scheme that can realize retrieval control, we slightly modify the typical PEKS scheme, which consists of the following six algorithms. The system framework includes three entities: 1) the sender; 2) the cloud; and 3) the recipient (and other users), whose specific functions are shown in Fig. 2.

1) *KeyGen*(λ) $\rightarrow$ (pk, sk): Input a security parameter λ and output the public-private key pair (pk,sk).
2) *PEKS*(pk, H) $\rightarrow$ I: Input the index matrix H and the receiver's public key pk, and output a searchable encrypted index I.
3) *Trapdoor*(sk, w_i) $\rightarrow$ T_{w_i}: Input a keyword $w_i \in W$ and the private key sk, and output a trapdoor T_{w_i}.

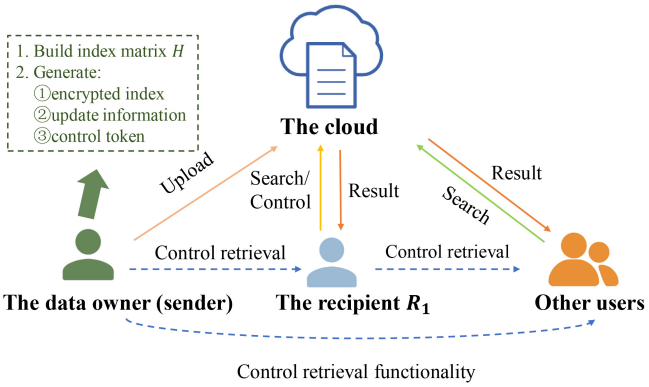


Fig. 2. System framework.

4) *Test*$(I, T_{w_i}) \rightarrow c_{w_i}$*:* Input the encrypted index I and the trapdoor T_{w_i}, and output the result set c_{w_i}.

Specifically, combined with auxiliary information, the owner constructs an index matrix H using file identifiers and keywords. Then, he encrypts H by using the recipient's public key pk to obtain the encrypted index I and sends the encrypted index to the cloud along with the encrypted data.

To make the scheme dynamic, we equip it with an *Update* protocol.

1) *Update*$(U, pk, I) \rightarrow I_{NEW}$*:* Input the update information U and the encrypted index I, and output the updated index ciphertext I_{NEW}.

In this phase, the data owner generates the update information U by keywords and identifier of the file to be updated and encrypts it under the public key pk. After receiving the ciphertexts, the server updates the encrypted index I. We call this "Noninteractive Update," i.e., the data owner does not have to implement the interactive "retrieve-decrypt-change-reencrypt-write back" protocol with the server.

In multiuser scenarios, to achieve controlled keyword search and file access, we also equip it with the *Control* protocol.

1) *Control*$(I, T_G) \rightarrow I_G$*:* Input the encrypted index I and the control token T_G, and output the controlled index ciphertext I_G.

B. Security Definition

Semantic Security: The basic security definition of typical PEKS is chosen-keyword attack (CKA) security. It requires that the adversary $\mathcal{A}$ cannot learn any information about the keyword from the encrypted index without seeing the related search trapdoor. In our dynamic PEKS scheme based on an inverted-index, we define *semantic security* for it based on *CKA* security definition. It ensures that $\mathcal{A}$ cannot learn any information about keywords from the index ciphertext and search results even if $\mathcal{A}$ can obtain the trapdoor of the keyword. Formally, *semantic security* is defined by the following *Game* between $\mathcal{A}$ and a challenger $\mathcal{C}$.

Step 1: The challenger $\mathcal{C}$ performs the following: Inputing a security parameter λ, $\mathcal{C}$ runs KeyGen$(\lambda) \rightarrow (pk, sk)$. Inputing the ordered keyword space $W = \{w_1, \ldots, w_m\}$ and the file identifiers set ID $= \{id_1, \ldots, id_h\}$, $\mathcal{C}$ runs $PEKS(pk, H) \rightarrow I$. Then, $\mathcal{C}$ sends pk, W, and ID to $\mathcal{A}$.

Step 2: $\mathcal{A}$ can adaptively make *Trapdoor* and *Test* queries to a challenger $\mathcal{C}$ for arbitrarily chosen keywords. $\mathcal{C}$ responds to the queries as follows:

1) *Trapdoor Query:* After receiving the trapdoor query for a keyword w_i from $\mathcal{A}$, $\mathcal{C}$ runs Trapdoor$(sk, w_i) \rightarrow T_{w_i}$ and sends T_{w_i} to $\mathcal{A}$.
2) *Test Query:* After receiving the search query for a trapdoor T_{w_i} from $\mathcal{A}$, $\mathcal{C}$ runs Test$(I, T_{w_i}) \rightarrow c_{w_i}$ and sends c_{w_i} to $\mathcal{A}$.
3) *Update Query:* After receiving the update information from $\mathcal{A}$, $\mathcal{C}$ uses it to update the encrypted index.

Step 3: $\mathcal{A}$ selects two keywords $w_0, w_1 \in W$ and sends to $\mathcal{C}$. $\mathcal{C}$ randomly selects a bit $b \in \{0, 1\}$, and generates trapdoor T_{w_b} for w_b. Then, $\mathcal{C}$ sends back to $\mathcal{A}$ the trapdoor T_{w_b} and the returned result c_{w_b} obtained by running $Test(I, T_{w_b}) \rightarrow c_{w_b}$.

Step 4: $\mathcal{A}$ can continue to adaptively make *Trapdoor*, *Test* and *Update* queries to $\mathcal{C}$ for arbitrarily chosen keywords as step 2.

Step 5: Finally, $\mathcal{A}$ outputs a guess b' that succeeds if $b' = b$.

The advantage of $\mathcal{A}$ in *Game* is defined as $\mathrm{Adv}^{\mathrm{Game}}_{\mathcal{A},DIIBPEKS}(\lambda) = |\Pr[b = b'] - 1/2|$.

Definition 1: A dynamic inverted-index-based PEKS scheme is *semantic security* iff the advantage of $\mathcal{A}$ is negligible in λ in the above game.

The search pattern reveals which queries are associated with the same keyword.

Theorem 1: If the adversary $\mathcal{A}$ outputs correct $b' = b$ with a negligible advantage, then it also implies satisfying search pattern privacy.

Proof: If the search pattern privacy is not satisfied, in step 3 of the *Game*, the adversary $\mathcal{A}$ can output correct $b' = b$ from the trapdoor T_{w_b} with nonnegligible advantage. Thus, *semantic security* implies search pattern privacy. ■

The access pattern reveals which queries return the same result.

Theorem 2: If the adversary $\mathcal{A}$ outputs correct $b' = b$ with a negligible advantage, then it also implies satisfying access pattern privacy.

Proof: If the access pattern privacy is not satisfied, in step 4, $\mathcal{A}$ issues a *Trapdoor* query for the keyword w_0 to obtain the trapdoor T_{w_0}. Then, $\mathcal{A}$ outputs correct $b' = b$ with a nonnegligible advantage based on the results returned by *Test* queries issued with the trapdoors T_{w_0} and T_{w_b}. Thus, *semantic security* implies access pattern privacy. ■

The size pattern reveals the volume of results matching the search trapdoor, as well as the number of keywords in the updated files.

Theorem 3: If the adversary $\mathcal{A}$ outputs correct $b' = b$ with a negligible advantage, then it also implies satisfying size pattern privacy.

Proof: If the size pattern privacy is not satisfied, the adversary $\mathcal{A}$ can output correct $b' = b$ with a nonnegligible advantage as follows:

In step 3, the adversary $\mathcal{A}$ can select and send to $\mathcal{C}$ two keywords $w_0, w_1 \in W$ which search results have different sizes. In step 4, $\mathcal{A}$ issues *Trapdoor* query for the keyword w_0 to get the trapdoor T_{w_0}. Then, $\mathcal{A}$ outputs correct $b' = b$ with a nonnegligible advantage based on the results returned by *Test* queries

issued with the trapdoors T_{w_0} and T_{w_b}. Therefore, *semantic security* implies size pattern privacy. ■

Forward privacy ensures that the newly added file containing the keyword w cannot be linked to the trapdoor generated for the same keywords w in previous search operations.

Forward security was first proposed in the construction of the Dynamic SSE schemes. If the DSSE schemes do not have forward security, the adversary can recover the keywords contained in files by the file injection attack [6]. Therefore, many researchers are studying how to construct a DSSE scheme that supports forward security, but this is not considered in typical PEKS schemes. In fact, the file injection attack is more likely to be applied in traditional PEKS scenarios. Because only the recipient's public key is required for a file injection attack, which is actually equivalent to a KGA. However, being resistant to KGAs does not imply forward security. This is because even if the adversary cannot actively conduct a file injection attack (as in the public key-authenticated SE scheme), the adversary can still judge whether the subsequent files have the same keywords as the previous search results based on the previous trapdoor. Therefore, when the keyword space is small, the adversary can infer the keyword information through prior knowledge (e.g., the search frequency) [6].

Backward privacy ensures that searches for the keyword w cannot be linked to the files containing the keyword w which have been deleted.

Theorem 4: If the adversary $\mathcal{A}$ outputs correct $b' = b$ with a negligible advantage, then it also implies satisfying forward privacy and backward privacy.

Proof: If the forward-backward privacy is not satisfied, in step 4, $\mathcal{A}$ issues an *Update* query for a file containing the keyword w_0. Then, $\mathcal{A}$ outputs correct $b' = b$ with a nonnegligible advantage based on the results returned by *Test* queries issued with the trapdoor T_{w_b}. Thus, *semantic security* implies forward privacy and backward privacy. ■

IV. Propose Scheme

A. Initial Design

It is known that the *Test* process is a linear transformation operation. However, the computation and communication costs of the retrieval process under the traditional HE schemes are intensive. To improve its practicability, we adopt IVHE.

In the index matrix, the bitmap of each row corresponding to a keyword is regarded as an integer, and all the integers form an integer vector which is encrypted via the public key of IVHE. A trapdoor for the keyword w_i is a random key-switching matrix generated according to $\mathbf{B} = [(0, \ldots, 0, 1, 0, \ldots, 0)]$ and the receiver's private key as described in Section II. According to the principle of a linear transformation, we obtain the search result by calculating the trapdoor and the ciphertexts of the integer vectors. At this time, the search result is the result of the linear transformation of matrix $\mathbf{B}$ and the integer vectors. After decrypting, the plaintext is regarded as a bit string where the *id* of the files containing the keyword w_i can be obtained. Furthermore, the randomness of the key-switching matrix makes the trapdoor generated for the same query matrix $\mathbf{B}$ random each time. Meanwhile, the ciphertext of the returned result is also random each time as it is the output of the linear transformation operation using the random key-switching matrix. That is, even the trapdoor and the result ciphertext of the same keyword are different and random. This makes our scheme have several good security features, such as hiding multiple patterns, etc.

In our scheme, it can update the file by homomorphic addition operation, and the size pattern (the number of keywords involved in the file to be updated) can be perfectly hidden in this process. It can also realize noninteractive update, which only requires a round of interaction between the client and the server, reducing information leakage.

However, we have to admit that when the element size or dimension of the vector is too large the initial idea is inefficient, so we have to further optimize this design.

B. Optimized Design

To support large databases, we utilize a packaging technique to handle the index settings to achieve reasonable computational and communication overheads. We set appropriate the vector element size and vector dimensions as needed. Specifically, we divide the rows into groups where each group is regarded as an integer of suitable size. Then, we form the integer vectors with suitable dimensions. In addition, the adversary may learn certain keywords of high query frequency through auxiliary information and then recovers the query keywords through the intersection operation in multiple queries [36]. To prevent this, inspired by [36], we evenly order the keywords according to the search frequency when constructing the index matrix, and store the sorted keywords in a table P. Finally, the index matrix is represented as a set of vectors with a relatively uniform frequency distribution.

We construct an optimized inverted-index-based PEKS scheme (*KeyGen*, *PEKS*, *Trapdoor*, *Test*, *Update*, *Control*) over the keyword space $\mathcal{W}$ as follows.

1) *KeyGen*(λ) $\rightarrow$ (pk, sk)*:* Taking a security parameter λ as input, and output public-private key pair (pk,sk) by running IVHE's key generation algorithm.
2) *PEKS*($PK, \{\vec{iv}\}$) $\rightarrow$ I*:* Take the IVHE's public key pk and the processed index vectors $\{\vec{iv}\}$ as input and output the encrypted index I. More concretely, let ID $= \{id_1, \ldots, id_h\}$ be the ordered shared file identifiers set, and $W = \{w_1, \ldots, w_m\}$ be the keywords set extracted from the shared files. Combined with the auxiliary information of different search frequencies of keywords, the data owner first constructs an index matrix H of size $m \times n$. The owner sets $H_{i,j} = 1$ if the jth file contains the keyword represented by the ith row, and $H_{i,j} = 0$ otherwise. Next, the owner divides the bit string of each row in the index matrix H into l_2 groups of the size of l_1 bits where $l_1 \times l_2 = n$, and each group is regarded as an integer. Now, the matrix can be regarded as a $m \times l_2$ integer matrix H_{int}. Then, each column of the integer matrix is divided into l_4 groups of the size of l_3, where $l_3 \times l_4 = m$, and each group is transformed into an integer vector of l_3 dimensions. Therefore, the index matrix is transformed into $l_4 \times l_2$ integer vectors $\{\vec{iv}\}$. Finally, each

Algorithm 1 *PEKS*

Input: The public key pk, ID, W
Output: The encrypted index I

Init Table P, two-dimensional array A
for the index matrix H of size $m \times n$ **do** ▷ Dependent on auxiliary information
  if $w_i \in id_j$ **then**
    $H_{(i,j)} = 1$
  else
    $H_{(i,j)} = 0$
  end if
end for
for $i = 1$ to m **do**
  Divide the bit string into l_2 groups of the size of l_1 bits which is regarded as integers ▷ Obtain integer matrix H_{int}
end for
for $j = 1$ to l_2 **do**
  each l_3 integers form a vector $\vec{iv}$
  $c_{sj} = IVHE.Enc(pk, \vec{iv})$
end for
for $i = 1$ to l_4 **do**
  for $j = 1$ to l_2 **do**
    $A[i][j] \leftarrow c_{ij}$ ▷ Send A to the server
  end for
end for

integer vector is encrypted via pk of IVHE's encryption algorithm to obtain the ciphertext set $\{c_{ij}\}$ of the integer vectors. An example is shown in Fig. 3(a). The data owner stores the ciphertext set in the 2-D array $A[i][j]$ where $1 \le i \le l_4, 1 \le j \le l_2$. See Algorithm 1 for details.

1) *Trapdoor*$(sk, w_i) \rightarrow T_{w_i}$*:* When the recipient wants to query the keyword w_i, he calculates the corresponding location of w_i in the vector by table P. First, the recipient queries the position p of w_i in table P and calculates the sequence number s and the corresponding position l through $s = 1 + (p-1)/l_3$ and $l = 1 + (p-1) \bmod l_3$. s represents the row number of the vector corresponding to w_i in the index array, and l represents the position of the element corresponding to w_i in the vector. Next, the recipient constructs a 1-D integer matrix $G_{1\times l_3}$ setting the lth element to 1, and the elements in other locations to 0. The recipient wants to perform a search operation in the ciphertext state. According to Section II, first, he randomly generates a new private key sk' based on the parameter settings of the matrix G. Then, he computes the key-switching matrix M using G, sk, and sk'. Finally, he sends M and the sequence number s to the server as a trapdoor T_{w_i}, refer to Algorithm 2.
2) *Test*$(I, T_{w_i}) \rightarrow c_{w_i}$*:* After receiving the trapdoor T_{w_i}, the server executes the linear transformation operation using M and all ciphertexts in the 2-D array $A[s][j]$, $1 \le j \le l_2$, and returns the result A' as the test result c_{w_i} to the recipient, refer to Algorithm 2.

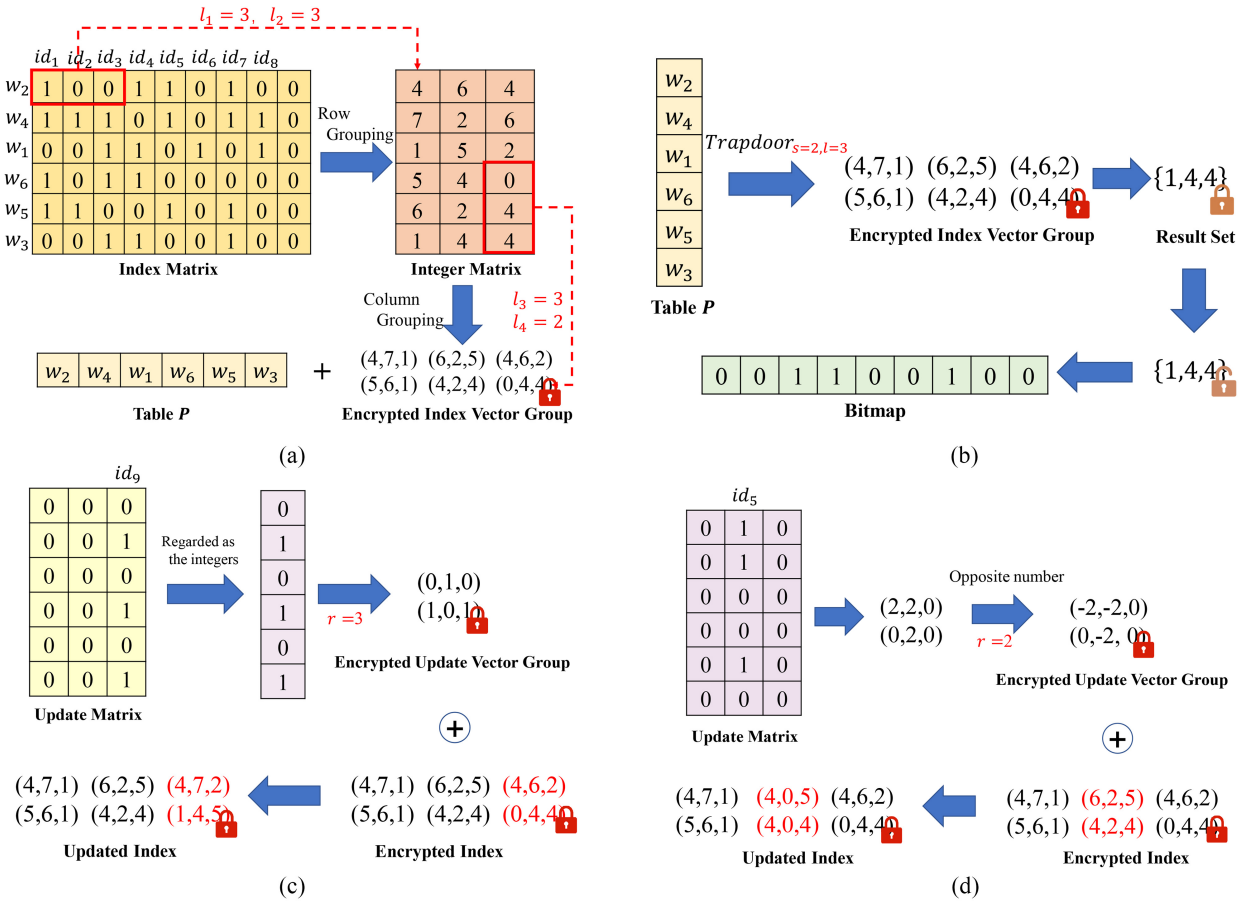
Algorithm 2 *Search* (*Trapdoor* − *Test*)

Input: w_i, (pk, sk), the encrypted index A
Output: The result identifiers $\{ID_{w_i}\}$

//Client:
Find p ▷ the position of w_i in table P
$s = 1 + (p-1)/l_3$ ▷ s represents the row number
$l = 1 + (p-1) \bmod l_3$ ▷ l represents the corresponding position in the vector
Init a one-dimensional zreo integer matrix $G_{1\times l_3}$
$Trapdoor(sk, w_i) \rightarrow T_{w_i}$
//Server:
for $j = 1$ to l_2 **do**
  $c_{w_j} = T_{w_i} \odot A[s][j]$ ▷ $\odot$ means linear transformation operation
  store $c_{w_j} \rightarrow A'$
end for
send A' to the client
//Client:
$\vec{ps} \leftarrow IVHE.Dec(sk, A')$
Form $\vec{ps}$ into bitmap bm in order (where each integer is treated as l_1 bits)

Adjustable Security: Different users or different fields may have different security requirements, such as sensitive databases needing high security, so we make a tradeoff between efficiency and security. In the above scheme, we match the trapdoor with the index ciphertext of the corresponding position (e.g., $A[s][j]$). The adversary can know the exact keyword to be searched with a small probability, but it has to be admitted that there is a certain leakage in this process.

Countermeasures: To achieve adjustable security, our remedy is to control the matching number of encrypted indexes and trapdoors in the *Test* phase. That is, the client can independently choose the number of the index to be matched according to the requirements of efficiency and security. For example, to achieve perfect search pattern hiding, we require trapdoors to match all ciphertexts in the encrypted index. Then, the matching results are stored in array A''. At this time, according to Section V, the computational cost of the search process is still acceptable (only 68 s for 1 million files), but the communication cost is relatively large. Therefore, to save communication overhead, we introduce a trusted execution environment (TEE) (e.g., Intel SGX). The TEE refers to a secure area built into the CPU through hardware and software methods. It guarantees the security, confidentiality, and integrity of the code and data loaded into that environment. In SGX, the environment in which an application runs is called an Enclave. SGX provides an attestation mechanism for the remote third party and Enclave, called SGX remote attestation. Through this technology, the client establishes a secure channel with the Enclave on the cloud server using the Diffie–Hellman key exchange protocol to achieve secret sharing. Therefore, the client achieves accurate filtering and transmission of search results through secure interaction with the Enclave. Specifically, the matching ciphertexts are read into the Enclave. According to the instructions of the client (i.e., the sequence number s), the



- Figure (a) indicates an example of index construction. Assuming that the index matrix is grouped by $l_1 = 3$ bits, each group is regarded as an integer, forming a $l_3 = 3$-dimensional vector. After encryption, an encrypted index and an auxiliary table P are obtained.
- Figure (b) shows the keyword search process. When searching for keyword w_3, the search trapdoor is first calculated according to table P, and then hand it over to the server for Test operation with A[s][j](i.e. A[2][j]).
- Figures (c)(d) indicate examples of an update operation. Based on the file to be updated (id_9 or id_5), the update matrix is constructed and the auxiliary parameter r is computed. The add and revoke operations are the same, except that the elements of the integer vector are taken as opposite numbers one by one in the revoke operation. The update ciphertext and parameter r are sent to the server to update the encrypted index.

Fig. 3. Illustration of operations. (a) Example of index construction. (b) Example of searching w_3 process. (c) Example of adding the file $id_9 = \{w_4, w_6, w_3\}$. (d) Example of deleting the file $id_5 = \{w_2, w_4, w_5\}$.

Enclave accurately filters out the result ciphertext corresponding to the keyword to be searched and sends them back to the client through the secure channel, where the computational and communication complexity are both $O(1)$. Note that the available memory size of Enclave is limited to (less than) 128 MB. Once exceeded, it will cause high-performance overhead.

After receiving the search result from the server, the recipient decrypts the ciphertext set with the new private key sk' to get the plaintext set ps. Each integer of ps is regarded as an l_1-bit string to which forms a bitmap bm. Finally, the recipient obtains the file identifiers $\{id_j\}$ containing the search keyword w_i by checking whether the jth bit in bm is 1. An example is shown in Fig. 3(b). When the recipient wants to query the keyword w_i again, he needs to recalculate a new search trapdoor (the key-switching matrix) M'.

1) *Update*: Our scheme supports updating operations to make itself dynamic.
Share/Add a New File: If the data owner wants to share a new file with the recipient, he proceeds as follows: He assigns to the new file an identifier $id_u \in ID$ which is not used, and then calculates the second dimension of the 2-D array (encrypted index) A corresponding to id_u as $r = 1 + (u-1)/l_1$. Then, he constructs a zero-bit matrix L of size $m \times l_1$. Suppose that the keywords set contained in the file id_u is W_u. If $w_i \in W_u$, the data owner finds the corresponding position p of w_i according to the table P and sets $L_{p,1+(u-1) \bmod l_1}$ to 1. Each row of L is regarded as an integer, and all the integers are divided in order into l_4 integer vectors of l_3-dimension. *Revoke/Delete a Shared File:* When the data owner wants to revoke a shared file from the recipient, he proceeds as follows: assuming that the file identifier is id_u, the data owner calculates the second dimension of the 2-D array (encrypted index) A corresponding to id_u as $r = 1 + (u-1)/l_1$. Then, he constructs a zero-bit matrix L of size $m \times l_1$. Suppose that the set of keywords contained in the file id_u is W_u. If $w_i \in W_u$, the data owner finds the corresponding position p of w_i according to the

table P and sets $L_{p,1+(u-1) \bmod l_1}$ to 1. Each row of L is regarded as an integer and then the integer is changed to its opposite number. Then, all the integers are divided in order into l_4 integer vectors of l_3-dimension.

These integer vectors are encrypted under the public key pk using IVHE, whose ciphertexts $uc[j]$ $(1 \le j \le l_4)$ and r are sent to the cloud server as the update token. Finally, the server performs a homomorphic addition operation between $uc[j]$ and $A[j][r]$ for $1 \le j \le l_4$, thereby realizing the addition/revoke of the file. The specific process is shown in Fig. 3(c) and (d).

It can be seen from the above update process that our scheme can also add and delete multiple files (corresponding files in the same vector) at the same time, i.e., batch update (partial), which can further improve the update efficiency. The batch update process is similar to the above process and will not be described in detail.

Expand Index: If a new shared file is assigned to the identifier id_{n+1}, i.e., the length of the bitmaps of keywords needs to increase. That is to say, the data owner needs to expand the encrypted index stored in the cloud server. Our scheme can expand the capacity of the encrypted index by $b \times l_1$ files when needed, which is not considered in the existing SE schemes using a bitmap. The data owner only needs to send to the cloud server the integer b and the ciphertext of the l_3-dimensional zero vector encrypted under the public key pk. Then, the server completes the index expansion by adding b zero vector ciphertexts at the end of each $A[i](1 \le i \le l_4)$.

Let's consider the following scenarios: A database is shared within the company, in which the search functionalities of the chairman and ordinary employees are different. The chairman has the authority to retrieve all data, while employees can only retrieve their related data. Similarly, in the medical systems, the doctors should be able to query all information about patients, such as disease/drug information, while patients can only access their own relevant information, and do not have the authority to retrieve the private information of other patients.

Similar scenarios are common in daily life. However, it is difficult to directly transfer the current PEKS schemes to achieve controlled retrieval authority under multiple users. In typical PEKS schemes, when the data owner $\mathcal{S}$ wants to share with multiple recipients U_k the search ability (Scenario 1), he must calculate the keyword ciphertexts for each file under the public key of each recipient. Especially, if the data owner $\mathcal{S}$ has already deleted the locally stored data, $\mathcal{S}$ must redownload all the data files and reextract the keywords for each file to calculate keyword ciphertexts under the public key of U_k. Consider another scenario: when a recipient $\mathcal{R}_1$ wants to share the data received from the data owner $\mathcal{S}$ with another recipient $\mathcal{R}_k$ (Scenario 2), the naive solution is also that $\mathcal{R}_1$ redownloads the data and recalculates the keyword ciphertexts under the public key of $\mathcal{R}_k$. In both scenarios, the common solution leads to heavy computing and communication overheads. "Proxy Re-encryption with keyword search" provides an alternative solution. It can transfer the keyword ciphertexts under the public key of $\mathcal{R}_1$ to that under the public key of $\mathcal{R}_k$ by the cloud server if needed. However, although there are some schemes proposed, they all need to transfer the keyword ciphertexts for all the data files one by one. They also have problems, such as no resistance to the KGA attacks [37], [38], [39], [40], deterministic trapdoor for the same keyword [37], [38], [39], [41], and using a designated tester [41].

Our scheme can solve these problems. To our knowledge, our scheme is the first PEKS scheme that can be directly extended to multiple users while ensuring high security. Since the owner can establish an isolation mechanism among multiple users by using only linear transformation operation, he does not need any trusted third party nor predetermine the users.

1) *Control:* Our scheme can realize multisearchable management.

Scenario 1: Data owner $\mathcal{S}$ constructs the encrypted index I via $PEKS(pk, W) \rightarrow I$ and sends it to the server. If $\mathcal{S}$ wants to share with recipients $U = \{U_1, U_2, \ldots\}$ the file access and the keyword search functionalities, the core is to build and store a new encrypted index I_k for each recipient $U_k \in U$ in the cloud. Since the key-switching matrix can change the key-ciphertext pair, the original index is switched to the index ciphertext under the new private key without revealing the keyword information. Specifically, $\mathcal{S}$ generates key pair (pk_k, sk_k) for U_k, switches the ciphertexts in I to the new ciphertexts under the public key pk_k in the cloud, and then gives the new private key sk_k to the recipient U_k.

① *Control the Search Functionality of Keywords:* During this process, the data owner $\mathcal{S}$ can control the keyword search functionality of each user $U_k \in U$, as follows:

The data owner $\mathcal{S}$ constructs a zero matrix $G_{l_3 \times l_3}$ (l_3 corresponds to the dimension of the integer vector introduced Section IV-B). Then, under the parameter settings of G, $\mathcal{S}$ randomly generates a new key pair (pk_{k_w}, sk_{k_w}) of the IVHE algorithm. By adjusting the elements in matrix $G_{l_3 \times l_3}$, $\mathcal{S}$ controls the keyword search authority of each user U: First, $\mathcal{S}$ finds out the keyword w_i to be authorized to the user U_k and judges the position l of the corresponding integer of w_i in the vector. Next, the element in the lth row and the lth column of the matrix G is set to 1. At this time, the integers corresponding to the unauthorized keywords in the result ciphertext obtained by linear transformation are 0.

The linear transformation process is as follows.

a) Based on the matrix G, the original private key sk and the new private key sk_{k_w}, the data owner S calculates the control token (key-switching matrix) T_{k_w} and sends it to the cloud.

b) The cloud generates a new encrypted index I_{k_w} for U_k by the token T_{k_w} and the corresponding ciphertext (i.e., the row(s) corresponding to the keyword(s) to be authorized) in the encrypted index, where I_{k_w} is the ciphertext under the new public key pk_{k_w}.

The data owner $\mathcal{S}$ can generate different G according to the keywords to be authorized, and then repeat the calculation process of linear transformation multiple times. Finally, $\mathcal{S}$ sends the new private key sk_{k_w} to the user U_k. Therefore, $\mathcal{S}$ can control the retrieval ability of other

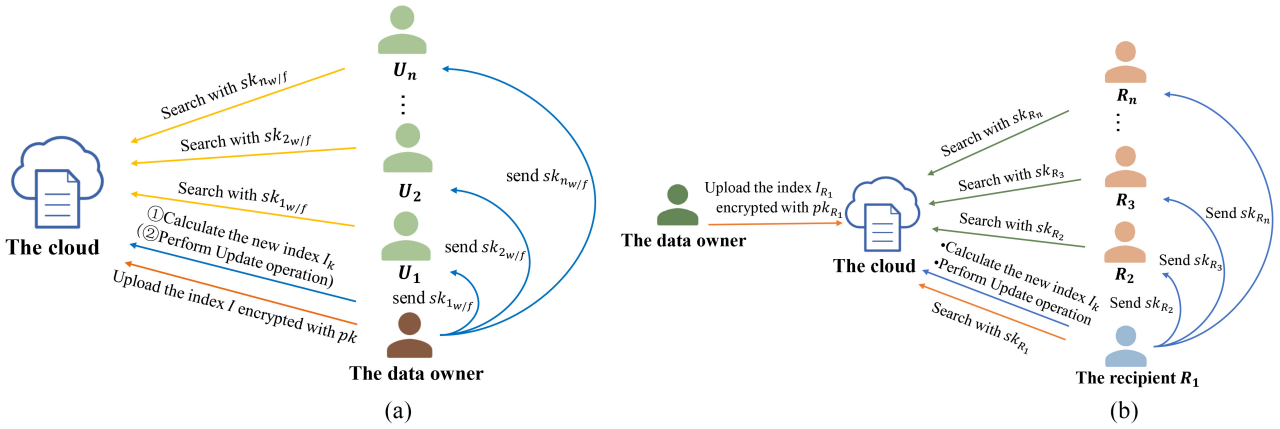


Fig. 4. Different scenarios. (a) Control the access authority of multiple recipients to files and keywords—Scenario 1. (b) Control the access authority of multiple recipients to files and keywords—Scenario 2.

recipients for his keywords without downloading the data files and the encrypted index. The user U_k can use the private key sk_{k_w} to generate trapdoors to search for keywords on I_{k_w}, but cannot access the initial encrypted index I. The process is shown in Fig. 4(a).

② *Control the Access Authority to Files:* When data $\mathcal{S}$ wants to control the access authority to the files by each user $U_k \in U$, the specific process is as follows.

The owner $\mathcal{S}$ constructs the matrix $G_{l_3 \times l_3}$, where G is the identity matrix. Similarly, $\mathcal{S}$ randomly generates a new key pair (pk_{k_f}, sk_{k_f}) of the IVHE algorithm. Based on the actual needs, $\mathcal{S}$ selects the corresponding columns (i.e., the columns corresponding to the files to be authorized) in the encrypted index and the matrix G to perform the linear transformation operation in the ciphertext state. As a result, a new encrypted index I_{k_f} is generated, which is based on the new public key pk_{k_f}.

The linear transformation process is as follows.

a) According to matrix G, the original private key sk and the new private key sk_{k_f}, S calculates the control token T_{k_f} which is sent to the cloud.

b) The cloud server generates a new encrypted index I_{k_f} for the user U_k by calculating the token T_{k_f} and the index part that the owner S wants to share.

If necessary, $\mathcal{S}$ can further delete/revoke certain files that he does not want to authorize in the encrypted index I_{k_f} by the *Update* operation. Finally, $\mathcal{S}$ sends the new private key sk_{k_f} to the user U_k. At this time, the user U_k can use the private key sk_{k_f} to generate trapdoors to search for keywords on I_{k_f}, but cannot access the initial encrypted index I. The process is shown in Fig. 4(a).

Scenario 2: Assume that the encrypted index built for the recipient $\mathcal{R}_1$ is I_{R_1}. $\mathcal{R}_1$ does the same operations as $\mathcal{S}$ does for U_k in Scenario 1. I.e., $\mathcal{R}_1$ can control other recipients' access authority to files and keywords through linear transformation and update operations. $\mathcal{R}_1$ generates key pair $(pk_{\mathcal{R}_k}, sk_{\mathcal{R}_k})$ for $\mathcal{R}_k$, builds the encrypted index I_{R_k} for $\mathcal{R}_k$ based on his own encrypted index I_{R_1}, and then gives the new private key $sk_{\mathcal{R}_k}$ to the recipient $\mathcal{R}_k$. $\mathcal{R}_k$ can use the private key $sk_{\mathcal{R}_k}$ to generate trapdoors to search for keywords on I_{R_k} rather than I_{R_1}. The process is shown in Fig. 4(b).

In our scheme, neither requiring a trusted third party nor predetermining the number of users, the data owner can securely control the access authority of other users to his data. That is, our scheme can be extended to a more demanding multiuser setting under fewer restrictions.

C. Proof of Security

The security of the proposed scheme depends on the security of the IVHE scheme. Please refer to [7], [33], and [34].

Theorem 5: Our scheme satisfies semantic security.

Proof: We analyze the semantic security of the scheme by a sequence of games.

Game$_0$: It is the same as *Game* except that in step 3, the challenger $\mathcal{C}$ obtains the returned result c_{w_b} as follows: Without loss of generality, assume that w_b is w_i. $\mathcal{C}$ first computes $l = 1 + (i-1) \bmod l_3$ and $h = (i-1)/l_3$. Then, $\mathcal{C}$ gets keywords set $W_r = \{w_{h \times l_3 + j}\}_{1 \le j \le l_3}$ where $w_{h \times l_3 + l} = w_i = w_b$, and generates the bitmap for each keyword in W_r. $\mathcal{C}$ builds a matrix using these bitmaps and divides each bitmap into l_2 integers which can be regarded as an $l_3 \times l_2$ integer matrix. Then, each column of the integer matrix is transformed into an integer vector of l_3 dimension. Thus, the matrix is transformed into $1 \times l_2$ vector array. Finally, the public key pk of IVHE is used to encrypt each vector in the integer vector array in order, thereby obtaining the ciphertext set $\{c'_t\}_{1 \le t \le l_2}$. Then, $\mathcal{C}$ replaces corresponding elements in the 2-D array A with $\{c'_t\}_{1 \le t \le l_2}$, i.e., $A[h+1][t] = c'_t$.

Observe that the process to generate the corresponding part of the encrypted index for W_r is a normal process in the PEKS algorithm of our scheme. Consequently, *Game$_0$* is the same as *Game*.

For $1 \le j \le l_3$, the following operations are available.

Game$_j$: is the same as *Game$_{j-1}$* except that the bitmap of $w_{h \times l_3 + j}$ is a random bit string.

We can obtain that $Game_{j+1}$ and $Game_j$ are indistinguishable for $0 \leq j \leq l_3 - 1$ by the semantic security of the IVHE scheme.

Finally, in $Game_{l_3}$, all the bitmaps corresponding to W_r are random bit strings. In our scheme, since the trapdoor (the key-switching matrix) generation algorithm is probabilistic, the generated search trapdoors are different even if search the same keyword. In addition, since the search frequency among index groups is indistinguishable, it is difficult for the adversary $\mathcal{A}$ to distinguish search trapdoor of w_b with the aid of auxiliary information. Similarly, according to the linear transformation calculation process, the returned results by the *Test* algorithm are also random and indistinguishable. Even if the same keyword is searched, the result ciphertexts are different. Therefore, the semantic security of IVHE prevents $\mathcal{A}$ from guessing the right challenge keyword by the test results with nonnegligible advantage. Since the number of identifiers corresponding to all keywords in the bitmap is the same, $\mathcal{A}$ cannot distinguish keywords based on the size of the result ciphertext. If the adversary $\mathcal{A}$ only has trapdoors, it can only distinguish w_b with negligible advantage. Likewise, even if the adversary $\mathcal{A}$ issues *Update* queries and changes the encrypted index, the result returned by the Test algorithm on the changed encrypted index is indistinguishable from the result returned by the Test algorithm on an encrypted index for the random bitmap matrix. Actually, even if the encrypted index is not changed, the results returned by the *Test* algorithm with different trapdoors are indistinguishable even if for the same keyword. Since all keywords participate in the update process, the adversary $\mathcal{A}$ cannot determine the number of keywords contained in the update file. Thus, the advantage of the adversary in $Game_{l_3}$ to output correct b' is negligible. ■

From the search pattern privacy and size pattern privacy, the adversary $\mathcal{A}$ cannot determine the specific search trapdoor and the specific keyword involved in the update process. Since it is difficult to link the trapdoor with the keyword to be updated, so the forward-backward security is realized.

In summary, the advantage of $\mathcal{A}$ winning the *Game* is negligible, so our scheme achieves adaptive semantic security.

Theorem 6: Our scheme can resist offline KGAs.

Proof: Our scheme constructs a (keyword, identifier bitmap) matrix and transforms it into integer vectors as the index. We use the key-switching matrix as the trapdoor and utilize the linear transformation operation of IVHE to realize the *Test* operation. Because there is no specific keyword ciphertext in our scheme, it can prevent the adversary from generating ciphertexts for known keywords to test a trapdoor as what the adversary does in KGA attacks for the typical PEKS schemes. In addition, the location of the real search result for a keyword is only known by the receiver itself and the search result plaintext must be obtained by decrypting the ciphertexts of the vectors set of the real search result using the receiver's private key. This guarantees that the adversary cannot determine which specific keyword is related to a trapdoor, thus achieving stronger security than the typical PEKS schemes. It completes the proof. ■

TABLE I
PERFORMANCE COMPARISON FOR DIFFERENT DIMENSIONS

	10	20	40	50
Construction of index (vectors/s)①	4000	1910	1040	588
Pre-computation (vectors/s)①	3225	1587	793	636
Trapdoor generation (ms)	0.018	0.1	0.13	0.24
Trapdoor (KB)	19.6	39.3	78.7	98.4

① $vectors/s$ represents the number of vectors running unit time

Different from the previous PEKS schemes resisting offline KGAs, our scheme neither needs a designed tester nor need to use private keys for signature.

Theorem 7: Our scheme achieves search pattern privacy, access pattern privacy, size pattern privacy, and forward and backward privacy.

Proof: From Theorems 1–5, we can straightly prove that our scheme achieves search/access/size pattern privacy, and forward and backward security. ■

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance concerning the computational and communication overheads of the *PEKS*, *Trapdoor*, *Test*, and *Update* operations, and analyze the additional overheads of *Control* protocol for the new application scenarios. We use the Numpy library in Python to implement the scheme on the Windows platform of Intel Core i7-8700 CPU @ 3.20 GHz 3.19 GHz, and 16.0 GB (15.8 GB is available) RAM. In the experiment, we modify the IVHE algorithm [7],[3] where the parameters $\ell = 63$ and $w = 2^{31}$ are set.

First, we determine the initial storage capacity of the index matrix as 1000×1240, which represents 1000 keywords and 1240 files.[4] To optimize efficiency as much as possible, we take the parameter $l_1 = 31$ bits, so each row in the index matrix is regarded as $l_2 = 40$ integers. Subsequently, the encrypted index can be expanded in real-time according to the files and keywords to be added. To further reduce the computational cost, the client has stored some zero-vector ciphertexts in advance. As a result, when the client requests to encrypt the zero vector, he does not need to encrypt it again but directly uploads the stored zero vector ciphertext.[5] Considering the special property of the IVHE algorithm, the ciphertext of a zero vector is still a zero vector. To address the problem, we replace the first element of the vector with a random 31-bit nonzero integer so that it is a nonzero integer vector.

Nextly, we evaluate the performance when the vector takes different dimensions, i.e., l_3 takes 10, 20, 40, and 50. Note that l_3 is restricted by the number of keywords (that is, m). All the reported costs are the average of 100 experiments. The results are shown in Table I.

[3]https://github.com/secureldata/Efficient-Integer-Vector-Homomorphic-Encryption

[4]This size is set to facilitate subsequent packing.

[5]Homomorphic addition is much faster than encryption. In the experiment, the client can continuously generate new zero vector ciphertext by homomorphic addition among the stored zero vector ciphertext to meet different needs.

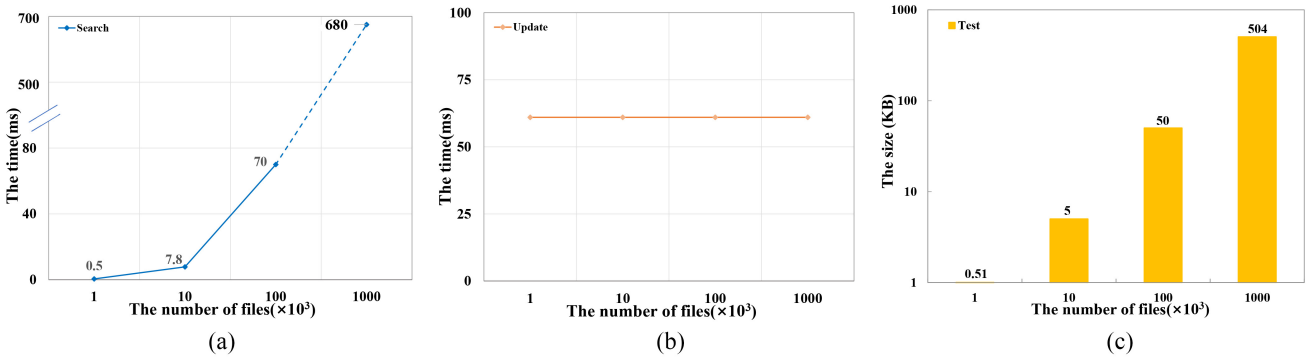


Fig. 5. Performance evaluation. (a) Computational cost of *Search* protocol. (b) Computational cost of *Update* protocol. (c) Communication overhead of *Search* protocol.

According to Section IV-B, the size and generation time of the trapdoor are independent of the number of files and keywords but are linear to the vector dimension. We can interpret the result as follows. In the *Test* process, it is required to perform linear transformation using the trapdoor and the vector ciphertexts in the encrypted index, so the computational cost of trapdoor generation is only dominated by the dimension of the query matrix, i.e., vector dimension l_3. When l_3 is fixed, the computational cost and the communication overhead of trapdoor generation remain constant. As shown in Table I, as the number of dimensions increases, the generation time also increases. After comprehensive consideration, we choose the parameter $l_3 = 10$ to continue the experiment. At this time, the initial index matrix can be indicated as 4000 vectors.

A. Computational Cost

Precalculation: To avoid redundant calculations, we require the server to perform some precalculation. That is, the server converts the index ciphertext c in A into the intermediate bit representation c^*. This operation is only performed once in the *Setup* stage and can accelerate the *Test* process. Therefore, this consumption is acceptable. Refer to Table I for the overhead of precomputation operations.

Test Phase: Unlike traditional PEKS schemes, in our scheme, we do not need to match the trapdoor to every keyword ciphertext in each file, so it can be applied to large databases. The specific cost is shown in Fig. 5(a) which includes testing and decrypting the results.

As shown in Fig. 5(a), the test time is linear with the number of files. The reason is that when the number of files increases, the number of vectors representing the index also increases. Therefore, more vector ciphertexts need to be linearly transformed with the trapdoor and then decrypted, which leads to a larger computational cost.

SGX Evaluation: In the tradeoff between efficiency and security, to save communication overhead and securely transmit the result ciphertext to the receiver, we introduce a TEE (e.g., SGX). Through the Intel SGX remote attestation mechanism, users can establish a secure channel with the Enclave on the cloud server using the Diffie–Hellman key exchange protocol, which takes 3.3 s. Since the process is only executed once during the entire system life cycle, we consider it acceptable.

Update Phase: From Section IV-B, the sender encrypts the update vectors[6] and sends them to the server. After receiving the ciphertexts, the server performs precomputation to convert them into the intermediate ciphertext (additional overhead), and then performs a homomorphic addition operation with the index ciphertext. Next, we consider the worst-case scenario where the update process does not involve zero-vector encryption, and the specific cost is 61 ms. As demonstrated in Fig. 5(b), the cost is independent of the number of files but is only related to the number of vectors involved in the keywords to be updated. Among them, the conversion overhead (additional overhead) is 31 ms.

B. Communication Cost

Since we adopt a unique index-building method and search method, we should analyze the specific communication overheads. In the *Test* phase, the communication overhead for the recipient to upload a search trapdoor is 19.6 kB, and the communication overhead of the cloud sending the search result to the recipient is shown in Fig. 5(c). It can be seen that the overhead is increasing linearly with the number of files. In addition, due to the introduction of SGX, when the client increases the number of encrypted index to be matched, the communication cost remains unchanged (same communication overhead as searching only for A[s][j]). In the *Update* phase, the sender uploads the ciphertexts of 100 integer vectors to the cloud server, for which communication overhead is 15.6 kB. According to Section IV-B, when the size of the keyword space remains unchanged, the communication overhead of the *Update* process also remains unchanged.

C. Cost of Control Protocol

In addition, we test the additional overheads of the *Control* protocol that implements controlled keyword retrieval in new application scenarios. It is known that traditional PEKS schemes cannot be directly extended to achieve multisearchable management. The *Control* protocol of our scheme perfectly solves this problem.

[6]Since the zero vector ciphertexts are stored in advance, when encryption of the zero vector is involved, the sender does not need to encrypt again.

TABLE II
COMPARISON OF COMPLEXITY ANALYSIS

	Build Inde/PEKS	Trapdoor generation	Search/Test	Comm_complexity (Search)
[12]	$2 \times m(\ell' \times \ell' \ C_E + \ell' \times \ell' \ C_{mul}) + mC_{sig}$	$2(\ell' \times \ell' \ C_E + \ell' \times \ell' \ C_{mul}) + C_{mul'} + C_{add} + C_{E'}$	$(m \times 2 \cdot \ell' + 1) \ C_p$	O $(2\ell' \cdot \lvert G \rvert + C_{sig} + \sigma)$
[4]	$3 \ C_E + C_H$	$C_E + C_H + C_p$	$2n \ C_p$ at best and $2mn \ C_p$ at worst	$O(\sigma)$
Our	$l_2 \times l_4 \ C_{enc}$	C_{ke-sw}	$l_2(C_{li-tr} + C_{dec})$	$O(n)$

The symbols C_E, C_{mul}, and C_H denote the complexity of modular exponentiation, multiplication operation on Bilinear group G and hash operation. The symbols C_{sig}, C_p, C_{add}, and $C_{mul'}$ denote the complexity of signature, pairing operation, addition operation, and multiplication operation in $\mathbb{Z}_q$, respectively. C_{enc}, C_{dec}, C_{ke-sw} and C_{li-tr} denote the complexity of encryption and decryption operations of IVHE, key switching matrix, and linear transformation operation. The symbol σ represents matching file information.

TABLE III
COMPARISON OF PERFORMANCE

	Build Index/PEKS	Trapdoor generation	Search/Test	Communication volume
[12]	When $\ell' = 3$, it takes 6 seconds①	6 milliseconds	6.001 seconds ①	about 1300 Byte ④
[4]	80 seconds at best and 800s at worst②	6 milliseconds	20 seconds at best and 200s at worst③	⑤
Our	When $\ell = 63$, $w = 2^{31}$, it takes 18.5 seconds.	18 microseconds	7.8 milliseconds (search+decrypt)	5×10^3 Byte

① The encryption time of file id information is not taken into account. It takes 1 millisecond to test a pairing operation.
② It takes 8 milliseconds to perform PEKS once. When constructing the index during Build Index, it is necessary to perform the PEKS operation on each keyword in the files, with a total of performing $\sum_{i=1}^{n} m_i$ PEKS (where n is the number of files, and m_i the number of keywords in the i-th file). In the best case, where the file contains only one keyword, it takes 80 seconds to perform 10000 PEKS operations. Assuming that a file has 10 keywords, it takes 800s in the worst case.
③ It takes 2 milliseconds to perform the Test once. When Searching, it is necessary to perform the Test operation on each keyword ciphertext in the files, and a total of performing $\sum_{i=1}^{n} m_i$ Test. In the best case, the Test is performed 10,000 times, which takes 20s. In the worst case, the Test is performed 100,000 times, which takes 200s.
④ If the file information is stored in bitmap format, the fixed communication overhead is 1300 bytes. When stored in file identifier format, a file identifier takes 2 bytes (not practical when matching files exceeding 700).
⑤ After testing the encrypted index, the results are not returned directly returned to the client but are notified to the server of the matching results.

TABLE IV
COMPARISON OF SECURITY PROPERTIES

	Forward/Backward Security	Search Pattern Privacy	Access Pattern	Size Pattern	Multi-Searchable Management	Non-Interactive Update
[12]	×	✓	×	×	×	×
[4]	×	×	×	×	×	✓
Our	✓	✓	✓	✓	✓	✓

We tested the computational cost and communication overhead of authorizing other users to query keywords or files. According to Section IV-B, combined with the specific needs, the computational cost of the user generating the control token is 0.59 ms, which is independent of the number of files and only related to the vector dimension. The control token is sent to the server, whose communication overhead is 196 kB. After receiving the control token, the server selects the corresponding encrypted index to perform a linear transformation operation with the actual needs. When $l_3 = 10$, the computational cost of this process is 22 000 (vectors/s).

Next, we compare the performance and security of our scheme with these of two typical schemes, including a PEKS scheme [12] based on the inverted index and an authenticated PEKS scheme [4] resistant to inside KGAs. Specifically, we have theoretically analyzed the computational and communication volumes of the three schemes, as shown in Table II. Due to the significant differences in the construction primitives between our scheme and [4], [12], we give the specific computational time and communication volume of each scheme under our parameter settings in Table III. We obtain their performances in the index construction, trapdoor generation, and search phases for 10 000 files and 1000 keywords. Table IV provides a comparison of security between solutions.

It can be seen that our solution has absolute advantages in security and computational cost which supports large databases with millions of files.

VI. INTEGRITY VERIFICATION OF SEARCH RESULT

In this section, we extend the above scheme to verify search results as follows.

In the $PEKS(pk, H) \rightarrow I$ algorithm, besides constructing the encrypted index I described in Section IV-B, a sequence of integers sqv is built for all keywords, where the integer corresponding to the keyword w represents the sum of the l_2 integers formed from the row in the matrix H corresponding to the keyword w. Next, the sequence of integers sqv is divided orderly into l_4 integer vectors of l_3-dimension for integrity verification of the results. For example, as shown in Fig. 3(a), the newly added l_3-dimensional verifiable integer vectors are required to be $\vec{V}_1 = (v_2, v_4, v_1)$ amd $\vec{V}_2 = (v_6, v_5, v_3)$, where $v_2 = 4+6+4 = 14$, $v_4 = 7+2+6 = 15$, $v_1 = 8$, $v_6 = 9$, $v_5 = 12$, and $v_3 = 9$. Then, these l_4 integer vectors are encrypted via the IVHE scheme, whose ciphertexts $vc[j]$ $(1 \leq j \leq l_4)$ called *verification vectors* are sent to the cloud along with I.

In the $\text{Test}(I, T_{w_i}) \rightarrow c_{w_i}$ algorithm, besides the ciphertexts in the 2-D array $A[s][j]$, $1 \leq j \leq l_2$, the trapdoor T_{w_i} also needs to execute the linear transformation calculation with the

ciphertext of each integer vector in the *verification vectors vc*. Then, the server returns the verification values along with the search result. The receiver gets and decrypts the ciphertext of the verification value corresponding to the searched keyword, and then compares the plaintext with the sum of the integers in the result set *ps*. If they are equal, the search result is complete and correct. Otherwise, it means that some errors occur in the returned result. As demonstrated in Fig. 3(b), the verification value obtained by decrypting the ciphertext of v_3 is equal to 9, and the sum of integers of the corresponding search result is $1+4+4=9$, which means that the returned result is complete.

In the *Update* process, in addition to updating the encrypted index, the ciphertexts $vc[j]$ $(1 \leq j \leq l_4)$ also need to be modified by performing homomorphic addition operation with the update vector ciphertexts $uc[j]$ $(1 \leq j \leq l_4)$ in order. As shown in Fig. 3(c), when the file id_9 is added, $\vec{V}_1 = (v_2, v_4, v_1)$ and $\vec{V}_2 = (v_6, v_5, v_3)$ are modified to be $\vec{V}_1 = (v_2+0, v_4+1, v_1+0)$ and $\vec{V}_2 = (v_6+1, v_5+0, v_3+1)$, i.e., $v_2 = 14$, $v_4 = 16$, $v_1 = 8$, $v_6 = 10$, $v_5 = 12$, and $v_3 = 10$. Similarly, as shown in Fig. 3(d), when the file id_5 is deleted, $\vec{V}_1 = (v_2, v_4, v_1)$ and $\vec{V}_2 = (v_6, v_5, v_3)$ are modified to be $\vec{V}_1 = (v_2-2, v_4-2, v_1+0)$, $\vec{V}_2 = (v_6+0, v_5-2, v_3+0)$, i.e., $v_2 = 12$, $v_4 = 13$, $v_1 = 8$, $v_6 = 9$, $v_5 = 10$, and $v_3 = 9$.

For implementation, it is verified that our integrity verification method of the returned result in databases of appropriate size is completely feasible and the extra overhead is small compared with the basic construction.

VII. Conclusion

In this article, we focus on how to improve the practicability and security of PEKS schemes in the scenario that a data owner outsources a large number of data files to the cloud server and shares them with multiple recipients. We design a dynamic inverted-index-based PEKS scheme utilizing index matrix structure and IVHE. Compared with the existing schemes, our construction has distinct advantages in security and computational cost. Using only the linear transformation operation, the data owner (the recipients) achieves multisearchable management, which increases the application value of PEKS. We also consider the verification of the search result which guarantees the correctness and completeness of the returned result and makes it supported by our scheme. To support more file identifiers, we have to consider some specific measures in designing the scheme, which may cause some additional costs, but we trade acceptable overhead for extremely high search efficiency. We believe that in the future, with the development of the IVHE technology, our scheme can continue to be optimized to achieve the goal of more efficiency.

References

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology (EUROCRYPT)*. Interlaken, Switzerland: Springer, 2004, pp. 506–522.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.

[3] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. NDSS*, 2012, pp. 1–15.

[4] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vols. 403–404, pp. 1–14, Sep. 2017.

[5] B. Qin, Y. Chen, Q. Huang, X. Liu, and D. Zheng, "Public-key authenticated encryption with keyword search revisited: Security model and constructions," *Inf. Sci.*, vol. 516, pp. 515–528, Apr. 2020.

[6] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. 25th USENIX Security Symp.*, 2016, pp. 707–720.

[7] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," in *Proc. ITA Workshop*, 2014, pp. 1–9.

[8] H. Yang, B. Jin, C. Chen, and X. Wu, "Efficient homomorphic encryption and its application," *J. Cryptol. Res.*, vol. 4, no. 6, pp. 611–619, 2017.

[9] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology (CRYPTO)*. Santa Barbara, CA, USA: Springer, 2001, pp. 213–229.

[10] M. Abdalla et al., "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Advances in Cryptology (CRYPTO)*. Santa Barbara, CA, USA: Springer, 2005, pp. 205–222.

[11] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *J. Syst. Softw.*, vol. 83, no. 5, pp. 763–771, 2010.

[12] R. Zhang, R. Xue, T. Yu, and L. Liu, "PVSAE: A public verifiable searchable encryption service framework for outsourced encrypted data," in *Proc. IEEE ICWS*, 2016, pp. 428–435.

[13] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Secure Data Management*. Berlin, Germany: Springer, 2006.

[14] B. Wang, T. Chen, and F. Jeng, "Security improvement against malicious server's attack for a dPEKS scheme," *Int. J. Inf. Educ. Technol.*, vol. 1, pp. 350–353, Jan. 2011.

[15] X. Pan and F. Li, "Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability," *J. Syst. Archit.*, vol. 115, May 2021, Art. no. 102075.

[16] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *Proc. IEEE INFOCOM*, 2015, pp. 2092–2100.

[17] Y. Wang, S.-F. Sun, J. Wang, J. K. Liu, and X. Chen, "Achieving searchable encryption scheme with search pattern hidden," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 1012–1025, Mar./Apr. 2022.

[18] C. Hu, P. Liu, R. Yang, and Y. Xu, "Public-key encryption with keyword search via obfuscation," *IEEE Access*, vol. 7, pp. 37394–37405, 2019.

[19] Q. Tang, "Nothing is for free: Security in searching shared and encrypted data," *IEEE Trans. Inf. Forensics Security*, vol. 9, pp. 1943–1952, 2014.

[20] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Data and Applications Security XXII*. London, U.K.: Springer, 2008, pp. 127–143.

[21] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Computer Security (ESORICS)*. Heraklion, Greece: Springer, 2016, pp. 173–195.

[22] D. Sharma and D. C. Jinwala, "Multiuser searchable encryption with token freshness verification," *Security Commun. Netw.*, vol. 2017, pp. 1–16, Nov. 2017.

[23] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing-Based Cryptography Pairing*. Tokyo, Japan: Springer, 2007, pp. 2–22.

[24] H. Wang, X. Dong, and Z. Cao, "Secure and efficient encrypted keyword search for multi-user setting in cloud computing," *Peer-to-Peer Netw. Appl.*, vol. 12, pp. 32–42, Jan. 2019.

[25] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," Apr. 2017, *arXiv:1704.03578*.

[26] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," Ph.D. dissertation, Dept. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 1978.

[27] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[28] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[29] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology (EUROCRYPT)*. Prague, Czech Republic: Springer, 1999, pp. 223–238.

[30] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. Symp. Theory Comput.*, 2009, pp. 169–178.
[31] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," *ACM Trans. Comput. Theory*, vol. 6, no. 3, pp. 309–325, Jul. 2014.
[32] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP," in *Advances in Cryptology (CRYPTO)*. Santa Barbara, CA, USA: Springer, 2012, pp. 868–886.
[33] C. Peikert, V. Vaikuntanathan, and B. Waters, "A framework for efficient and composable oblivious transfer," in *Advances in Cryptology (CRYPTO)*. Santa Barbara, CA, USA: Springer, 2008, pp. 554–571.
[34] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proc. STOC*, 2005, pp. 84–93.
[35] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. FOCS*, 2011, pp. 97–106.
[36] C. Liu, L. Zhu, M. Wang, and Y.-A. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Inf. Sci.*, vol. 265, pp. 176–188, May 2014.
[37] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Inf. Sci.*, vol. 180, no. 13, pp. 2576–2587, 2010.
[38] L. Fang, W. Susilo, C. Ge, and J. Wang, "Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search," *Theor. Comput. Sci.*, vol. 462, pp. 39–58, Nov. 2012.
[39] Z. Chen, S. Li, Q. Huang, Y. Wang, and S. Zhou, "A restricted proxy re-encryption with keyword search for fine-grained data access control in cloud storage," *Concurrency Comput. Pract. Exp.*, vol. 28, no. 10, pp. 2858–2876, 2016.
[40] Y. Yang, X. Zheng, V. Chang, and C. Tang, "Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage," *Concurrency Comput. Pract. Exp.*, vol. 29, no. 19, p. e4211, 2017.
[41] W. Zhang, B. Qin, X. Dong, and A. Tian, "Public-key encryption with bidirectional keyword search and its application to encrypted emails," *Comput. Stand. Interfaces*, vol. 78, Oct. 2021, Art. no. 103542.

Guiyun Qin received the master's degree from Shandong University, Qingdao, China, in 2023.

Her research interests include searchable encryption, cloud data security, and ciphertext database.

Pengtao Liu received the master's degree from Shandong University, Jinan, China, in 2006.

She is currently a Professor with the School of Cyberspace Security, Shandong University of Political Science and Law, Jinan. Her main research interests include searchable encryption and leakage-resilient cryptography.

Chengyu Hu received the Ph.D. degree from Shandong University, Jinan, China, in 2008.

He is currently an Associate Professor with the School of Cyber Science and Technology, Shandong University, Qingdao, China. His main research interests include cloud system security, public key cryptography, and network security.

Zengpeng Li received the Ph.D. degree from Harbin Engineering University, Harbin, China, in 2017.

He is currently an Associate Researcher with the School of Cyber Science and Technology, Shandong University, Qingdao, China. He has worked on lattice-based cryptography, password-based cryptography, and cryptographic protocol.

Shanqing Guo received the Ph.D. degree from Nanjing University, Nanjing, China, in 2006.

He is currently a Professor with the School of Cyber Science and Technology, Shandong University, Qingdao, China. His main research interests include network security and software security.

Information Sciences 520 (2020) 15–30

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Enabling cloud storage auditing with key-exposure resilience under continual key-leakage

Chengyu Hu[a,b,c,*], Yuqin Xu[d], Pengtao Liu[e], Jia Yu[f], Shanqing Guo[a,b], Minghao Zhao[g]

[a] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, 250100, China
[b] School of Cyber Science and Technology, Shandong University, Qingdao, 266237, China
[c] Key laboratory of Network Assessment Technology, CAS (Institute of Information Engineering, Chinese Academy of Sciences), Beijing, 100093, China
[d] School of Software, Shandong University, Jinan, 250101, China
[e] College of Cyberspace Security, Shandong University of Political Science and Law, Jinan, 250014, China
[f] College of Computer Science and Technology, Qingdao University, Qingdao, 266071, China
[g] School of Software, Tsinghua University, Beijing, China

ARTICLE INFO

Article history:
Received 7 April 2019
Revised 2 February 2020
Accepted 5 February 2020
Available online 5 February 2020

Keywords:
Data storage
Cloud storage auditing
Continual key-leakage resilience
Forward security

ABSTRACT

Cloud storage auditing is a service that is usually provided to enable clients to verify the integrity of their data stored in the cloud. However, clients risk exposing their secret key. To address the problem of key exposure, researchers have provided "Forward Security" by dividing the entire lifetime of the secret key into several periods and updating the secret key within each of these periods. Forward security can ensure the validity of authenticators before the period in which the secret key is fully exposed. However, the security of these protocols can be broken by launching side-channel attacks to leak the secret key partially rather than fully. In this study, we focus on implementing measures in cloud storage auditing to protect against side-channel attacks in practice. We formalize the definition and security model of a cloud storage auditing protocol, which supports forward security under continual key-leakage, and construct the first protocol. Our protocol remains secure even if an adversary obtains partial leakage of the secret key during a period. In addition, if the secret key were to be fully disclosed in a certain period, our protocol would maintain forward security. Therefore, the proposed protocol provides stronger security compared with existing protocols.

1. Introduction

Cloud storage is an emerging technology that provides clients with convenient data-related services. Recently, many world-leading IT companies have released cloud storage products, such as Google Cloud Storage, Microsoft Azure, and Amazon S3. Clients who utilize these cloud storage services rent the storage capacity and network bandwidth in a pay-as-you-go manner. Accordingly, they can outsource their data to the cloud and access the data anytime, anywhere through the internet, and enjoy other storage services based in the cloud (e.g., data analysis or image processing) if required. This obviates

* Corresponding author at: School of Cyber Science and Technology, Shandong University, Qingdao, 266237, China.
E-mail address: hcy@sdu.edu.cn (C. Hu).

https://doi.org/10.1016/j.ins.2020.02.010
0020-0255/

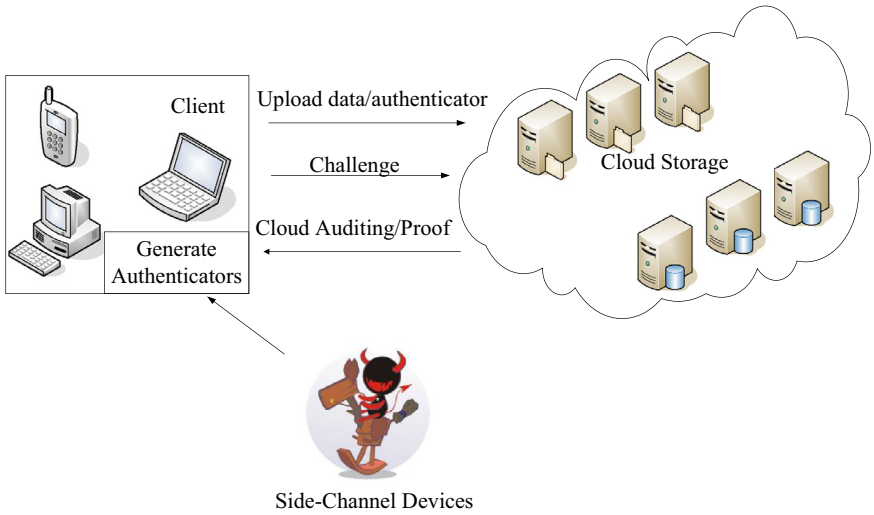


Fig. 1. System model of our cloud storage auditing.

the need for clients to maintain basic storage infrastructure, and the storage service provider can concentrate on the quality of the service themselves. Many individuals and institutions have adopted cloud storage to maintain their data. Since their inception, cloud storage services have become a lucrative industry, with the global cloud storage market estimated to reach $65.41 billion by the year 2020 [1].

Despite the numerous advantages of using a cloud storage service, data integrity has always been a significant problem that has prevented prospective clients from adopting this service. When users upload their data to the cloud, they lose complete control of their data and rely entirely on the cloud to maintain them. Although cloud service providers adopt a variety of advanced techniques (eg. replication [2] or erasure code [3]) to ensure data reliability and robustness, data corruption still frequently occurs [4]. In addition, a dishonest cloud server may conceal the incident of data loss to the users, or even maliciously delete users' data. Accordingly, from a user's perspective, the service provider should convince the user that the data they saved in the cloud will remain intact.

Unfortunately, unlike traditional settings in which hash functions and signatures can be utilized for integrity insurance, in a cloud storage scenario, the clients seldom retain a local copy of their data. In addition, it would be unrealistic to require the clients to download the entire dataset. Thus, it is necessary to compose an appropriate integrity auditing mechanism in a cloud storage scenario that can remotely verify the intactness of the data without reliance on local copies [5,6]. In this regard, cloud storage auditing protocols are cryptographic protocols that can efficiently and effectively prove the intactness of the data stored in the cloud. They normally adopt a spot-checking technique and thus the auditors are only required to access a fraction of the data to verify the integrity of the entire dataset. Consequently, cloud storage auditing has become a tool of significant importance for cloud data security.

Most of the cloud auditing protocols that have been developed to date assume that the client's secret key for auditing is securely maintained. However, in practice, newly emerging side-channel attacks may invalidate this assumption. Traditional techniques that were used to launch side-channel attacks such as a power analysis attack [7,8], timing attack [9], and electromagnetic analysis [10] were expensive to carry out and sometimes led to observable physical damage to the affected device. Modern side-channel attacks (eg. [11–13]) can grab users' secret key inexpensively and imperceptibly. For example, as shown by Genkin [14], who jointly analyzed different traces (e.g., the far end of a cable, human touch, electromagnetic, and power consumption), it is feasible to extract 4096-bit RSA keys and 3072-bit ElGamal keys from laptops with little effort. Once the secret key for auditing is leaked to the cloud service provider, all the present cloud auditing protocols would fail.

Schemes concentrating on secret key leakage in cloud auditing have been proposed [15–17], with all of them addressing the problem of cloud auditing as a result of *key disclosure*. These schemes consider the client's key to be fully leaked rather than partially leaked in a side-channel attack. For example, the client may inadvertently and carelessly download malware that reads the client's key and sends it to the attacker. In these previous studies, the entire lifetime of the secret key was separated into several time periods and *forward security* for a cloud auditing protocol was provided by updating the secret keys among the periods. As a result, these auditing protocols still remain secure in those periods that occur before the secret key is fully exposed. In practice, however, the adversary can obtain pieces of information about the secret key between two updates by launching a side-channel attack, which can obviously help it breach the security of the auditing protocol.

In this study, we focus on enabling leakage-tolerant cloud storage auditing to overcome the problem of partial key leakage between two key updates in the forward-secure cloud auditing protocols. Specifically, our proposed cloud storage auditing method achieves both "forward security" and "key-leakage resilience" simultaneously. Fig. 1 shows the scenario on which our work is based. The two participants are: the client (file owner) and the cloud. The client partitions each of his

files to blocks and uploads the blocks and the corresponding authenticators to the cloud. The client can use a service based in the cloud to verify whether their files are correctly stored in the cloud. An adversary can obtain partial information about the client's secret key by using side-channel attacks.

In this regard, leakage-resilience has attracted considerable attention in theoretical cryptography as an algorithmic countermeasure (contrary to engineering countermeasures such as hiding [18] and masking [19]) against side-channel attacks. In leakage-resilient cryptography, leakage models are generalized to capture the features of multiple types of side-channel attacks. Among these models, the *continual memory leakage model* is generally considered to be the most powerful model, which assumes the secret key in the memory can be (partially) acquired by the adversary.

1.1. Contribution

To make the auditing protocol support both of "forward security" and "key-leakage resilience" simultaneously, we first propose an auditing protocol with continual key-leakage resilience. Then, we extend the scheme to achieve our goal. The main contributions are as follows:

1. First, we attempted to provide the storage auditing protocol with continual key-leakage resilience, a capability previous auditing protocols did not have. Our design enables malicious operations on the client's cloud data to be detected, even if the malicious cloud obtains partial information about the client's current secret key for cloud storage auditing. We define *continual key-leakage resilience* for the cloud auditing protocol and propose the first concrete protocol for cloud storage.
2. We developed a cloud storage auditing protocol to support "*forward security*" and "*continual key-leakage resilience*" simultaneously. This protocol makes it possible to detect malicious operations on the client's cloud data in previous time periods, even if the malicious cloud server were to obtain the client's current secret key for cloud storage auditing and partial information about the secret keys of previous time periods. Specifically, we employ a binary tree structure [20,21] to update the clients secret keys in different time periods. We apply an existing technique [20] to our continual key-leakage resilient auditing protocol and propose the first auditing protocol with the above-mentioned two security properties.

1.2. Related work

Data Auditing for cloud storage. Remote data integrity verification has its origins in integrity protection memory management systems [22], which enable a client to verify whether read/write operations are correctly executed in unreliable memory. With the proliferation of cloud storage, proof of retrievability (POR) [23] and proof of data possession (PDP) [5,24] were proposed to efficiently verify the integrity of archival datasets. Specifically, a POR scheme stores each encrypted file in the cloud server along with a set of pseudorandom blocks. Subsequently, the client can examine the data integrity by verifying whether the server retains the pseudorandom blocks. PDP follows a different approach by allowing the client to verify the integrity by challenging the server with some randomly selected block numbers to determine whether the server generates valid proofs.

Later, multiple PDP and POR schemes were proposed to extend the performance or functionality of traditional schemes. For example, dynamic PDP [6,25,26] enables the client's file archive to be dynamically updated (e.g., via file upload or delete). PDP or POR with public verifiability (e.g., [27–29]) enables a third party, rather than the client, to verify the data integrity. Other solutions (e.g.,[30–32]) took privacy into consideration and ensured that neither the cloud nor the auditor could acquire the user's data.

The aforementioned studies (including ours) adopted the *single-server* model, which regards the cloud storage platform as a whole entity. Accordingly, they only focus on integrity verification in the cloud but cannot recover the original data when an inconsistency is found. It is worth mentioning that another approach was to adopt the *multi-server* model with the aim of reconstructing the compromised data by using a redundancy (e.g., replication or coding) technique. For example, a replication technique was adopted for data-recovery [33], whereas the high-availability and integrity layer (HAIL) [2] utilizes erasure coding, and a third approach involved regenerating codes in recovering corrupted data [34,35].

Leakage-resilient cryptographic protocols for the cloud. Secure multiparty computation (SMPC) [36,37] is a generic cryptographic protocol that enables distributed parties to jointly compute a functionality, while ensuring that each party's input and output remains secret. Generally, SMPC first transforms the targeted functionality into arithmetic or logic circuits for subsequent evaluation in a secure manner. Theoretically, the goal of leakage resilience SMPC is to secure circuit evaluation against an adversary who probes the values of internal wires. Several researchers (e.g., [38–40]) conducted in-depth research in this field.

Likewise, secret sharing [41] is a kind of cryptographic protocol that enables a user to randomly split a secret into multiple shares, such that certain subsets of the shares can be used to reconstruct the secret and others do not reveal any particulars of the secret. Secret sharing is also a significant tool for constructing secure cloud applications [42]. The leakage resilience of secret sharing was formalized by the work of Benhamouda et al. [43], after which several leakage resilient secret sharing schemes were proposed [44,45]. In terms of application-level secure cryptographic schemes for cloud computing, Hu et al. [46] and Dai et al. [47] considered leakage resilience for searchable encryption [48] to enable secure search in the cloud.

Studies that are the most closely related to this one are [15,16], all of which focused on the problem of cloud auditing *under key disclosure*. However, as mentioned previously, these solutions only provide "forward security" and do not consider the problem of partial key leakage between two key-updates.

1.3. Organization

In Section 2, we introduce the necessary preliminaries. Then, in Section 3, we propose a concrete auditing protocol with continual key-leakage resilience and analyze its security and performance. In Section 4, we extend the protocol in Section 3 such that it supports "forward security" and "continual key-leakage resilience" simultaneously. Finally, we conclude the paper in Section 5.

2. Preliminaries

2.1. Composite order bilinear groups

Our protocols are constructed on the composite order bilinear groups of order N where $N = p_1p_2p_3p_4$ is a product of four distinct primes [49]. Let G, G_T be cyclic groups of order N. Let e: $G \times G \to G_T$ be a map satisfying the following properties:

1. Bilinearity: For all $u, v \in G$ and any $a, b \in Z_N$, $e(u^a, v^b) = e(u, v)^{ab}$;
2. Non-degeneracy: For all generators $g \in G$, $e(g, g) \neq 1_{G_T}$;
3. Computability: $e(u, v)$ can be computed efficiently for all $u, v \in G$;

Following the explanation in [49], the composite order bilinear groups have some properties. Let G_{abc} be the subgroup of order abc for $a, b, c \in \{1, p_1, p_2, p_3, p_4\}$. There is an isomorphism $G_{mn} \cong G_m \times G_n$ if gcd(m,n)=1. Let G_{p_i} be subgroups of order p_i. Any element of G is in the form of $g_{p_1}^{x_1} g_{p_2}^{x_2} g_{p_3}^{x_3} g_{p_4}^{x_4}$ and $\prod_{i\in S} g_{p_i}^{x_i}$ is an element of subgroup $G_{\prod_{i\in S} p_i}$, where $S \subseteq \{1, 2, 3, 4\}$, g_{p_i} is a generator of subgroup G_{p_i} and $x_i \in Z_{p_i}$. The orthogonal property guarantees that if u, v are group elements of different order, then $e(u, v) = 1_{G_T}$.

In [49], some assumptions were proposed which can be used to prove the security. The security of our protocol mainly relies on the following assumption. Let $I = (N = p_1p_2p_3p_4, G, G_T, e)$ be a random bilinear setting.

Assumption 1: Pick $g_1, U_1 \leftarrow G_{p_1}$, $U_2, V_2 \leftarrow G_{p_2}$, $V_3, g_3 \leftarrow G_{p_3}$, $g_4 \leftarrow G_{p_4}$, $C_1 \leftarrow G_{p_1p_2p_3}$, $C_2 \leftarrow G_{p_1p_3}$ and set $E = (I, g_1, g_3, g_4, U_1U_2, V_2V_3)$. The advantage of an algorithm $\mathcal{A}$ to break Assumption 1 is defined to be

$$\mathrm{Adv}^{\mathcal{A}}_{Asmp1}(\lambda) = |\Pr[\mathcal{A}(E, C_1) = 1] - \Pr[\mathcal{A}(E, C_2) = 1]|.$$

It is proved in [49] that the above assumption holds in the generic group model, i.e., for all PPT algorithms $\mathcal{A}$, $\mathrm{Adv}^{\mathcal{A}}_{Asmp1}(\lambda)$ is a negligible function in λ.

2.2. Signature

A signature scheme typically consists of three algorithms (KeyGen,Sig,Ver). To ensure it becomes continual key-leakage resilient, it should be equipped with an additional algorithm KeyUpdate as follows:

1. KeyGen(1^λ) $\to$ (vk, sk_0): The key generation algorithm takes as input a security parameter λ and outputs the public verification key/private signing key pair (vk, sk_0).
2. Sig(sk_i, m) $\to \sigma$: The signing algorithm takes as input a private signing key sk_i and a message m, and outputs a signature σ.
3. Ver(vk, m, σ) $\to$ 0/1: The verification algorithm takes as input the public verification key vk, a message m, and a signature σ. If the output is 1, σ is a valid signature of m; otherwise the output is 0.
4. KeyUpdate(sk_{i-1}) $\to sk_i$: The key update algorithm takes as input the private signing key sk_{i-1}. It outputs a re-randomized key sk_i for the same verification key. sk_i has the same length as sk_{i-1} and a distribution that is indistinguishable from that of sk_{i-1}.

***l*-Leakage resilience security.** The security of "*l-leakage resilience against continual leakage on memory and computation* (CLR)" for signatures has been defined [49]. However, this definition of the signature [49] requires the signing algorithm to first update the signing key to a new one, and then sign the message. In our definition, this requirement is unnecessary. We define the security of *l-continual leakage resilience* (CLR) for our signatures based on the following game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

1. Setup phase. The challenger $\mathcal{C}$ takes a security parameter 1^λ and executes the KeyGen algorithm to return the verification key vk to $\mathcal{A}$ while keeping the signing key sk_0 itself. Set $i = 1$. Let L_{sk} be the numbers of leaked bits with the signing key in the current time period. No leakage is allowed in this phase.
2. Query phase. The adversary $\mathcal{A}$ adaptively issues the following three kinds of queries: *Signing Queries*. $\mathcal{A}$ supplies a message m to $\mathcal{C}$. The challenger signs the message and returns the resulting signature.
   Leak queries. Let $sk = sk_{i-1}$ be the current signing key. $\mathcal{A}$ supplies to $\mathcal{C}$ a polynomial-time computable arbitrary function f: $\{0, 1\}^* \to \{0, 1\}^*$, and receives $f(sk)$ or $\perp$ from $\mathcal{C}$ depending on whether the amount of leaked bits exceeds the

leakage bound, where sk is the signing key in the current time period. In addition, $\mathcal{C}$ updates the number of leaked bits with sk by adding $|f(sk)|$ or 0 to it. *Update Queries*. $\mathcal{A}$ asks the challenger C to update the signing key. C updates the signing key from sk_{i-1} to sk_i. Set $i = i + 1$.

3. Forgery phase. The adversary $\mathcal{A}$ supplies the challenger with a message/signature pair, (m^*, σ^*), with the restriction that m^* has not been previously queried. The adversary wins the game if Ver(vk, m^*, σ^*)=1.

Definition 1. A signature scheme SIG=(KeyGen,Sig,Ver,KeyUpdate) is *l-continual leakage resilient*(CLR) if the advantage $\mathrm{Adv}^{CLR}_{SIG,\mathcal{A}}(\lambda)$ of any probabilistic polynomial-time adversary $\mathcal{A}$ to win the above game is a negligible function in λ.

A signature scheme with *l-leakage resilience security*, which is also *l-continual leakage resilient* under the above definition, was developed [49]. We describe the scheme as follows and show that multiple signatures can be aggregated to ensure that it satisfies the requirement of the auditing protocol.

1. KeyGen(λ) $\rightarrow$ (vk, sk_0). It chooses a composite order bilinear group G as described in Section 2.1. It randomly selects $g, u, h \leftarrow_R G_{p_1}$ and $R, R', R'', R''' \leftarrow_R G_{p_4}$. The verification key is set to be $vk = \{N, G, R, gR', uR'', hR'''\}$. Then it randomly selects $g_2 \leftarrow G_{p_2}$, $g_3 \leftarrow G_{p_3}$, and random vectors $\vec{r} = (r_1, \cdots, r_n), \vec{c} = (c_1, \cdots, d_n), \vec{d} = (d_1, \cdots, d_n), \vec{f} = (f_1, \cdots, f_n), \vec{x} = (x_1, \cdots, x_n), \vec{y} = (y_1, \cdots, y_n), \vec{z} = (z_1, \cdots, z_n) \in Z_N^n$. Denote $g^{\vec{r}}$ to be the vector with n group elements $(g^{r_1}, \cdots, g^{r_n})$ and $g^{\vec{r}}g_2^{\vec{c}}$ to be the vector with n group elements formed by componentwise multiplication $(g^{r_1}g_2^{c_1}, \cdots, g^{r_n}g_2^{c_n})$. We let $\vec{S_0} = (S_{1,0}, \cdots, S_{n,0}), \vec{U_0} = (U_{1,0}, \cdots, U_{n,0})$ and $\vec{H_0} = (H_{1,0}, \cdots, H_{n,0})$ be vectors with n group elements defined as follows:
$\vec{S_0} = g^{\vec{r}}g_2^{\vec{c}}g_3^{\vec{x}}$, $\vec{U_0} = u^{\vec{r}}g_2^{\vec{d}}g_3^{\vec{y}}$, $\vec{H_0} = h^{\vec{r}}g_2^{\vec{f}}g_3^{\vec{z}}$
The signing key is $sk_0 = \{\vec{S_0}, \vec{U_0}, \vec{H_0}\}$ (this contains $3n$ group elements).
2. Sig(m, sk_i) $\rightarrow \sigma$. The signing algorithm produces the signature σ under current signing key sk_i as:
$\sigma = (\sigma_1, \sigma_2) = (U_{1,i}^m H_{1,i}, S_{1,i})$
where *name* is arbitrary here and can be treated as the file identifier when this scheme is used in auditing protocol.
3. Ver(vk, m, σ) $\rightarrow$ {"True" or "False"}. The verification algorithm checks that
$e(\sigma_1, gR'') = e(\sigma_2, (uR'')^m(hR''')) \neq 1$, and $e(\sigma_1, R) = e(\sigma_2, R) = 1$.
4. KeyUpdate(sk_{i-1}) $\rightarrow sk_i$.

The signature scheme presented above can be used to generate authentication values in the auditing protocol. The structure of the signature allows multiple signatures to be aggregated into a linear combination as follows. We must point that, to make the aggregation possible, blocks of a file identified by *name* should be signed under the same signing key sk_i.

Aggregation. Suppose there are n message/signature pairs $\{(m_1, \sigma_1), \cdots, (m_n, \sigma_n)\}$. We can construct a query which is an s-element set $Q = \{(i, v_i)\}$ by first randomly choosing an s-element subset I of $[1, \cdots, n]$ which denotes the block indexes and then choosing an element v_i for each $i \in I$. We can aggregate the signatures by computing the value $\mu \leftarrow \Sigma_{i \in I} v_i m_i$ and an aggregated signature $\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2) = (\prod_{i \in I} \sigma_{1,i}^{v_i}, \sigma_{2,*})$, where $\sigma_{1,i}$ is the first part of σ_i. Note that the second part of σ_i equals that of σ_j, we denote $\sigma_{2,*}$ as the second part of all the σ_i for all $i \in I$. The aggregated signature can be verified by checking that $e(\tilde{\sigma}_1, gR'') = e(\tilde{\sigma}_2, (uR'')^{\mu}(hR''')^{\sum_{i \in I} v_i}) \neq 1$, and that $e(\tilde{\sigma}_1, R) = e(\tilde{\sigma}_2, R) = 1$.

3. Auditing protocol with continual key-leakage resilience

In this section, we propose our first auditing protocol and prove that it achieves continual key-leakage resilience security. We also present an analysis of the performance of our protocol.

3.1. Definition and security model

To resist side-channel attacks, the client's secret key for auditing should be updated periodically while leaving the public key unchanged. Therefore, an auditing protocol with continual key-leakage resilience security consists of the following five algorithms (SysSetup, AuthGen, ProofGen, ProofVerify, KeyUpdate):

- SysSetup(1^{λ}) $\rightarrow$ (PK, SK_0): the system setup algorithm is executed by the client and takes as input a security parameter λ, and generates a public key PK and the client's initial secret key SK_0.
- AuthGen(PK, SK_{i-1}, F) $\rightarrow$ (Φ): the authenticator generation algorithm is executed by the client and takes as input the public key PK, the client's current secret key SK_{i-1} and a file F, and generates the set of authenticators Φ for F.
- ProofGen(PK, $Chal$, F, Φ) $\rightarrow$ (P): the proof generation algorithm is run by the cloud server and takes as input the public key PK, a challenge $Chal$ which is randomly selected by the client and sent to the cloud, a file F and the set of authenticators Φ, and generates a proof P that the cloud has correctly preserved F.
- ProofVerify(PK, $Chal$, P) $\rightarrow$ ("True" or "False"): the proof verification algorithm is executed by the client to verify the proof generated by ProofGen(PK, $Chal$, F, Φ). It takes as input the public key PK, the same challenge $Chal$ used in ProofGen and the proof P, and outputs "True" or "False".
- KeyUpdate(SK_{i-1}) $\rightarrow$ (SK_i): the key update algorithm is executed by the client and takes as input the client's current secret key SK_{i-1}, and returns a re-randomized secret key SK_i with the same length as SK_{i-1} and a distribution that is indistinguishable from that of SK_{i-1}.

Security model. We consider the leakage resilience security [49] and data possession property [50] in the security model. An adversary (possibly the cloud itself) can obtain partial information about the client's secret key for auditing between two key-update operations. This means that key leakage may occur in the model. The security of *l-continual leakage resilience*(CLR) for the auditing protocol is based on the following game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

1. Setup phase. The challenger $\mathcal{C}$ takes a security parameter λ and implements the SysSetup algorithm to return the public key PK to $\mathcal{A}$ while retaining the client's initial secret key SK_0 itself. Set $i = 1$. Let L_{SK} be the number of leaked bits with the current secret key. No leakage is allowed in this phase.
2. Query phase. The adversary $\mathcal{A}$ adaptively issues the following queries:
   Authenticator queries. $\mathcal{A}$ adaptively selects and sends a series of blocks $m_1, \cdots, m_n$ to the challenger $\mathcal{C}$. $\mathcal{C}$ computes and sends to $\mathcal{A}$ the authenticators for $m_k (k = 1, \cdots, n)$ under the current secret key. $\mathcal{A}$ stores all blocks $F = (m_1, \cdots, m_n)$ and their corresponding authenticators.
   Leak queries. Let $SK = SK_{i-1}$ be the current secret key. $\mathcal{A}$ supplies to $\mathcal{C}$ a polynomial-time computable arbitrary function f: $\{0, 1\}^* \to \{0, 1\}^*$, and receives $f(SK)$ or $\perp$ from $\mathcal{C}$ depending on whether the number of leaked bits exceeds the leakage bound, where SK is the client's current secret key. Then, $\mathcal{C}$ updates the number of leaked bits with SK by adding $|f(SK)|$ or 0 to it.
   Update queries. $\mathcal{A}$ asks the challenger $\mathcal{C}$ to update the secret key SK_{i-1}. $\mathcal{C}$ updates the secret key from SK_{i-1} to a re-randomized secret key SK_i with the same length as SK_{i-1} and a distribution that is indistinguishable from that of SK_{i-1}.
3. Challenge phase. The challenger $\mathcal{C}$ sends $\mathcal{A}$ a challenge *Chal* and asks $\mathcal{A}$ to provide a proof of the correct preservation of the blocks $m_{s_1}, \cdots, m_{s_c}$ of file $F = (m_1, \cdots, m_n)$ under *Chal*, where $1 \le s_l \le n$, $1 \le l \le c$, and $1 \le c \le n$.
4. Forgery phase. The adversary $\mathcal{A}$ outputs a proof P for the preservation of the blocks $m_{s_1}, \cdots, m_{s_c}$ under *Chal*. $\mathcal{A}$ wins the game if ProofVerify(*PK, Chal, P*)="True".

Definition 2 (*Continual Key-leakage Resilience*)**.** An auditing protocol is *l-continual leakage resilience*(CLR) if for any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage of the adversary to win the above game is a negligible function in λ.

Definition 3. (*Detectability*) [15]: An auditing protocol is (ρ, δ)-*detectable* ($0 < \rho, \delta < 1$) if the probability to detect a fraction ρ of corrupted blocks is at least δ.

3.2. Proposed protocol

Let Π be the above signature scheme in Section 2.

- SysSetup(1^λ) $\to$ (*PK, SK*$_0$): Let λ be the security parameter. It runs Π.KeyGen(λ) and sets $PK = \Pi.vk$, $SK_0 = \Pi.sk_0$. Let $Hash : \{0, 1\}^* \to G_{p_1}$ be a cryptographic hash function.
- AuthGen(PK, SK_{i-1}, F) $\to$ (Φ): Let the file $F = \{m_1, \cdots, m_n\}$ be identified by *name*, where $m_i \in Z_N (i = 1, \cdots, n)$. The client first computes $\{\sigma_i\}_{1 \le i \le n}$, where $\sigma_i = \Pi.Sig(m_i, SK_{i-1})$. Then the client computes $U = g^r g_2^{r_2} g_3^{r_3}$ where r, r_2, $r_3 \in Z_N$ is randomly selected. The authenticator $\Phi = (U, \{(\sigma_{1,i} \cdot Hash(name \parallel i \parallel U)^r \cdot u^{rm_i}, \sigma_{2,i})\}_{1 \le i \le n})$
- ProofGen(*PK, Chal, F,* Φ) $\to$ (*P*): The client randomly selects and sends to the cloud a challenge $Chal = \{(i, v_i)\}_{i \in I}$, where $I = \{s_1, \cdots, s_c\}$ is a c-element subset of set $[1, n]$ and $v_i \in Z_N$. Let file $F = \{m_1, \cdots, m_n\}$ be identified by *name*. The cloud computes an aggregated authenticator $\Phi = (U, \tilde{\sigma})$, where $\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2) = (\prod_{i \in I} \sigma_{1,i}^{v_i} \prod_{i \in I} Hash(name \parallel i \parallel U)^{rv_i} \prod_{i \in I} u^{rm_i v_i}, \sigma_{2,*})$ as above. It also computes the linear combination of randomly selected blocks $\mu \leftarrow \Sigma_{i \in I} v_i m_i$. It then sends $P = \{\Phi, \mu\}$ to the client along with the file tag as the response proof of correct file storage.
- ProofVerify(*PK, Chal, P*) $\to$ ("True" or "False"): Let the challenge *Chal* be $\{(i, v_i)\}_{i \in I}$. By receiving a proof *P*, the client checks whether the following equations hold:
  $e(\tilde{\sigma}_1, gR'') = e(\tilde{\sigma}_2, (uR'')^\mu (hR''')^{\sum_{i \in I} v_i}) \prod_{i \in I} e(Hash(name \parallel i \parallel U)^{v_i}, U) e(u^\mu, U) \neq 1$,
  and
  $e(\tilde{\sigma}_1, R) = e(\tilde{\sigma}_2, R) = 1$.
  If they all hold, it returns "True"; otherwise it returns "False".
- KeyUpdate(SK_{i-1}) $\to$ (SK_i): It runs $SK_i \leftarrow \Pi.KeyUpdate(SK_{i-1})$ to update the secret key.

Correctness. The ProofVerify algorithm returns "True" if the valid proof *P* is generated under the random challenge *Chal*. Note that for any secret key SK_i, the G_{p_1} parts of $\vec{S_i}, \vec{U_i}, \vec{H_i}$ have the form of $g^{\vec{r'}}, u^{\vec{r'}}, h^{\vec{r'}}$ for some $\vec{r'} \in Z_N^n$. Thus, $\sigma_{1,i}$ can be written as $\sigma_{1,i} = (u^{m_i} h)^{r'} g_2^{s_2} g_3^{s_3}$, and $\sigma_{2,i}$ can be written as $\sigma_{2,i} = g^{r'} g_2^{t_2} g_3^{t_3}$ for some value r', s_2, s_3, t_2, $t_3 \in Z_N$. Then:

$e(\tilde{\sigma}_1, gR'') = e(\prod_{i \in I} \sigma_{1,i}^{v_i}, gR'') e(\prod_{i \in I} Hash(name \parallel i \parallel U)^{rv_i}, gR'') e(\prod_{i \in I} u^{rm_i v_i}, gR'') = e((u^\mu h^{\sum_{i \in I} v_i})^{r'}, g) \prod_{i \in I} e(Hash(name \parallel i \parallel U)^{v_i}, U) e(u^\mu, U) \neq 1$

$e(\tilde{\sigma}_2, (uR'')^\mu (hR''')^{\sum_{i \in I} v_i}) = e(g^{r'} g_2^{t_2} g_3^{t_3}, (uR'')^\mu (hR''')^{\sum_{i \in I} v_i}) = e((u^\mu h^{\sum_{i \in I} v_i})^{r'}, g) \neq 1$ i.e. $e(\tilde{\sigma}_1, gR'') = e(\tilde{\sigma}_2, (uR'')^\mu (hR''')^{\sum_{i \in I} v_i}) \prod_{i \in I} e(Hash(name \parallel i \parallel U)^{v_i}, U) e(u^\mu, U)$. and

$e(\tilde{\sigma}_1, R) = e(\tilde{\sigma}_2, R) = 1$.

Table 1
Basic information.

Ellipse Curve Type	Type A1
Ellipse Curve	$y^2 = x^3 + x$
Symmetry or not	Symmetry
Order	$N = p_1 p_2 p_3 p_4$
Security Level	$\log p_i$=192
Platform	Personal Computer
CPU Series	Intel Core i5-6300
RAM	8GB
Operate System	Windows 10
JDK Version	JDK 1.8.0
jPBC Version	2.0.0

3.3. Security analysis

Theorem 1 (Continual Key-leakage Resilience)**.** *If **Assumption 1** holds, then the above auditing protocol is continual key-leakage resilient.*

Proof. First, as the signature scheme on which our auditing protocol is based is *l-continual leakage resilient*, the leakage in the authenticator generation algorithm cannot improve the advantage of the adversary to break the security of our auditing protocol. Next, we show that, if the adversary can compute a proof P for the blocks $m_{s_1}, \cdots, m_{s_c}$ integrity under *Chal*, and pass the ProofVerify check, the challenger can break **Assumption 1** with a non-negligible advantage. □

Suppose the forged proof for the query $Q = \{(i, v_i)\}$ is $P' = \{U', \tilde{\sigma}', \mu'\}$ and the expected valid proof generated by an honest prover is $P = \{U, \tilde{\sigma}, \mu\}$ where $\tilde{\sigma}.\tilde{\sigma_1} = \prod_{(i,v_i)\in Q} \sigma_{1,i}^{v_i} \prod_{(i,v_i)\in Q} Hash(name \parallel i \parallel U)^{rv_i} \prod_{(i,v_i)\in Q} u^{rm_i v_i}$ and $\mu = \sum_{(i,v_i)\in Q} v_i m_i$.

As $\tilde{\sigma}'$ can pass the ProofVerify check, i.e.

$e(\tilde{\sigma}_1', gR'') = e(\tilde{\sigma}_2', (uR'')^{\mu'}(hR''')^{\sum_{(i,v_i)\in Q} v_i}) \prod_{(i,v_i)\in Q} e(Hash(name \parallel i \parallel U')^{v_i}, U') e(u^{\mu'}, U')$, and $\prod_{(i,v_i)\in Q} e(Hash(name \parallel i \parallel U')^{v_i}, U') e(u^{\mu'}, U')$ only comes from $e(\tilde{\sigma}_1', gR'')$, the $\tilde{\sigma}_1'$'s G_{p_1} part from the signature Π should be equal to that of $\tilde{\sigma_1}$. That is, $u^{\mu'} h^{\sum_{(i,v_i)\in Q} v_i} = u^{\mu} h^{\sum_{(i,v_i)\in Q} v_i} \bmod N$, i.e. $u^{\mu'} = u^{\mu} \bmod N$. Note that $\mu \neq \mu' \bmod N$ which means that $\mu \equiv \mu' \bmod p_1$. Now, we can compute $a = gcd(\mu' - \mu, N)$ and $b = N/a$ satisfying that one of a, b is equal to p_1. Without loss of generality, let $a = p_1$ and $b = p_2 p_3 p_4$. The challenger can break **Assumption 1** by checking whether $e((U_1 U_2)^a, C)$ equals 1. As $(U_1 U_2)^a = U_2^a \bmod N$, if $e((U_1 U_2)^a, C) = 1$, which means C does not contain part of G_{p_2}, then $C \in G_{p_1 p_3}$; otherwise, $C \in G_{p_1 p_2 p_3}$.

Theorem 2 (Detectability)**.** *Our auditing protocol is* $(\frac{t}{n}, 1 - (\frac{n-t}{n})^c)$ *detectable if the cloud stores a file with n blocks, and deletes or modifies t blocks.*

Proof. According to the definitions, n blocks of the file are stored in the cloud, including t corrupted blocks (by deletion or modification). Therefore, if the challenged blocks selected randomly by the client contain at least one corrupted block, then the corrupted blocks can be detected. The probability that a randomly selected block is not among the corrupted t blocks is $1 - \frac{t}{n}$. Therefore, none of c randomly selected blocks in the corrupted t blocks is $(1 - \frac{t}{n})^c$. Now, we can obtain the detectable probability as $1 - (1 - \frac{t}{n})^c = 1 - (\frac{n-t}{n})^c$. □

3.4. Performance evaluation

In this section, we present a performance analysis of our protocol. We implement our protocol based on jPBC Library [http://libeccio.di.unisa.it/projects/jpbc/]. Table 1 summarizes the basic information of our implementation.

The order of the group has $192 \times 4 = 768$ bits. Accordingly, the sizes of an element in Z_N and G are 96 bytes and 196 bytes, respectively. We divide the data file into 1,000,000 blocks, which is approximately 91.55M bytes.

We demonstrate the duration of authenticator generation in Fig. 2 by varying the number of file blocks. We also illustrate the duration of the challenge generation, the proof generation, and the proof verification for a different number of checked data blocks in Fig. 3.

Focusing on the communication messages in our auditing protocol, we evaluate the size of the challenge and the proof messages in bytes in the proof generation process. Fig. 4 shows the linearity of the size of the challenge message with the number of checked blocks. Fig. 5 indicates that the size of the proof message is constant, i.e., 488 B.

4. Extension to forward secure protocol under continual key-leakage

In practice, the client's secret key of the auditing protocol may be fully exposed. Usually, clients prefer to use software-based key management to manage their different keys for different security goals. The limitation of software-based key management and careless mistakes by the client make it possible for the key to be exposed. In addition, if data loss incidents

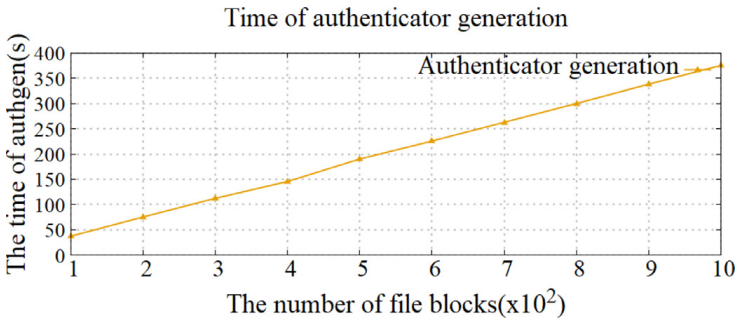


Fig. 2. The time of authenticator generation with different number of blocks.

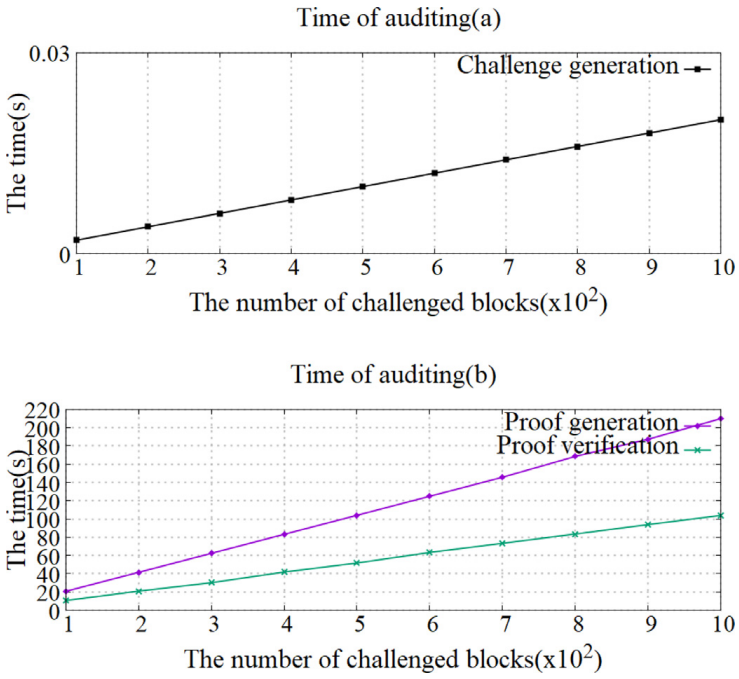


Fig. 3. The time of auditing procedures with different number of checked blocks.

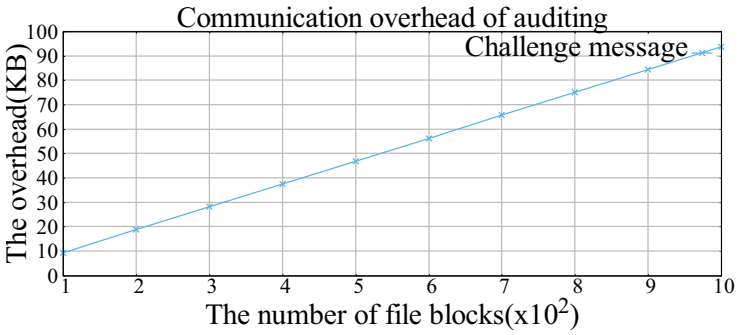


Fig. 4. Communicational Cost. (a) The size of the challenge message with different number of checked blocks.

were to occur on the cloud server side or, for storage cost reasons, the cloud server discards data the client rarely accesses, then the cloud service provider might want to obtain the clients secret keys for auditing purposes to forge authenticators and conceal the fact. Full exposure of their auditing secret key would clearly be disastrous for a client of cloud storage applications. Therefore, solving this problem to prevent exposure of the client's auditing secret key is of critical importance. However, to the best of our knowledge, although auditing protocols, which provide forward security under key exposure by updating the auditing secret key periodically, have been developed [15–17], they do not consider the severity of the consequences caused by partial secret key leakage between key-updates as a result of side-channel attacks. In this section, we describe the extension of our auditing protocol (as described in Section 3) with security against continual key-leakage attacks such that it supports both "key-exposure resilience" and "continual key-leakage resilience" simultaneously.

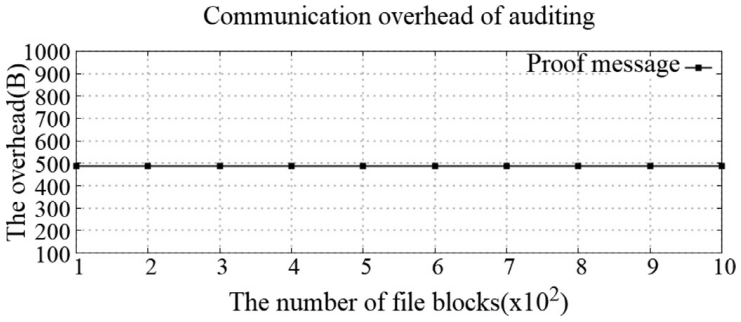


Fig. 5. Communicational Cost. (b) The size of the proof message with different number of checked blocks.

A method to securely construct a forward-secure signature scheme under continual key-leakage from any signature scheme under continual key-leakage has been described [20]. In this section, we apply their tree-based construction method to the continual key-leakage resilient auditing protocol we present in Section 3 and construct a forward-secure auditing protocol under continual key-leakage. In the following subsections, we first present the "forward security under continual leakage" or "key-exposure resilience under continual leakage" definition for auditing protocol, and then show the concrete construction of our protocol.

4.1. The model

To equip an auditing protocol with forward security (key-exposure resilience) under continual leakage, we modify an existing auditing protocol model with forward security [15]. An auditing protocol with forward security under continual leakage consists of the following five algorithms (SysSetup, AuthGen, ProofGen, ProofVerify, KeyUpdate):

- SysSetup(1^λ, T) $\rightarrow$ (PK, $SK_{0,0}$): The system setup algorithm is executed by the client and divides the entire lifetime into T periods. It takes a security parameter λ and T as input and outputs a public key PK and the client's initial secret key $SK_{0,0}$. Note that $SK_{j,i}$ is denoted as the current secret key in time period j which has been updated i times in this time period, i.e. the secret key in a time period can be updated to a new one for the same time period to resist continual key-leakage.
- AuthGen(PK, j, $SK_{j,i-1}$, F) $\rightarrow$ (Φ): The authenticator generation algorithm is executed by the client and takes as input the public key PK, a time period j, a client's current secret key $SK_{j,i-1}$ and a file F, and generates the set of authenticators Φ for F.
- ProofGen(PK, j, $Chal$, F, Φ) $\rightarrow$ (P): The proof generation algorithm is executed by the cloud and takes as input the public key PK, a time period j, a random challenge $Chal$ selected by the client and sent to the cloud, a file F, and the set of authenticators Φ, and generates a proof P for the clouds correct preservation of F.
- ProofVerify(PK, j, $Chal$, P) $\rightarrow$ ("True" or "False"): The proof verification algorithm is executed by the client to verify the proof generated by ProofGen(PK, j, $Chal$, F, Φ). It takes as input the public key PK, a time period j, the same challenge $Chal$ used in ProofGen and a proof P, and returns "True" or "False".
- KeyUpdate(PK, j, $SK_{j,i-1}$) $\rightarrow$ ($SK_{j+1,0}$): The key update algorithm is executed by the client and takes as input the public key PK, the current time period j and the client's current secret key $SK_{j,i-1}$, and outputs a new secret key $SK_{j+1,0}$ for the next time period $j+1$.

Forward security(key-exposure resilience) under continual leakage. The security definition of forward security under continual leakage for the auditing protocol is based on the following game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

1. Setup phase. Set time period $j=0$. The challenger $\mathcal{C}$ takes a security parameter λ and implements the SysSetup algorithm to return the public key PK to $\mathcal{A}$ while retaining the client's initial secret key $SK_{j,0}$. Set $i=1$. Let L_{SK} be the number of leaked bits with the current secret key in time period j. No leakage is allowed in this phase.
2. Query phase. The adversary $\mathcal{A}$ adaptively issues the following queries:
   Authenticator queries. $\mathcal{A}$ adaptively selects and sends a series of blocks $m_1, \cdots, m_n$ to the challenger $\mathcal{C}$. $\mathcal{C}$ computes and sends to $\mathcal{A}$ the authenticators for $m_k (k=1,\cdots,n)$ under the current secret key in time period j. $\mathcal{A}$ stores all blocks $F=(m_1,\cdots,m_n)$ and their corresponding authenticators.
   Leak queries. Let $SK=SK_{j,i-1}$ be the current secret key in time period j. $\mathcal{A}$ supplies to $\mathcal{C}$ a polynomial-time computable arbitrary function f: $\{0,1\}^* \rightarrow \{0,1\}^*$, and receives $f(SK)$ or $\perp$ from $\mathcal{C}$ depending on whether the amount of leaked bits exceeds the leakage bound, where SK is the client's current secret key. Then, $\mathcal{C}$ updates the number of leaked bits with SK by adding $|f(SK)|$ or 0 to it.
   Update queries. This query models the key update in period j for the resilience of continual key-leakage. $\mathcal{A}$ asks the challenger $\mathcal{C}$ to update the secret key $SK_{j,i-1}$ in time period j. $\mathcal{C}$ updates the secret key from $SK_{j,i-1}$ to a re-randomized secret key $SK_{j,i}$ with the same length as $SK_{j,i-1}$ and a distribution indistinguishable from that of $SK_{j,i-1}$.

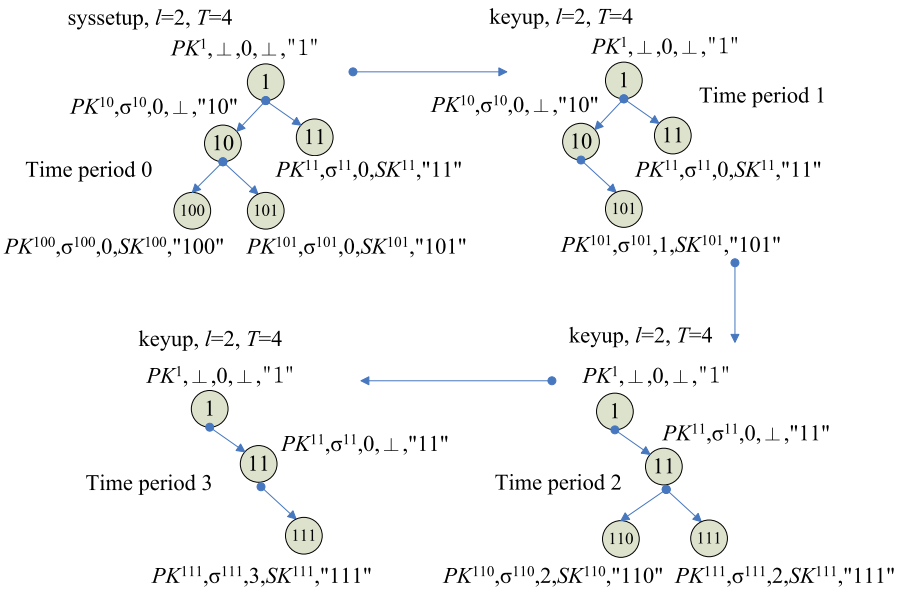


Fig. 6. Structure of thebinary tree in the lifetime.

At the end of each time period j, $\mathcal{A}$ can decide to remain in the query phase for the next time period $j = j + 1$ or proceed to the break-in phase.

3. Break-in phase. In this phase, the adversary $\mathcal{A}$ sets the break-in time period $b = j$, which means that key exposure occurs during this time period. The challenger $\mathcal{C}$ generates the secret key $SK_{b,0}$ by using the KeyUpdate algorithm and sends it to $\mathcal{A}$.
4. Challenge phase. The challenger $\mathcal{C}$ sends $\mathcal{A}$ a challenge *Chal*, a time period $j^*(j^* < b)$ and asks $\mathcal{A}$ to provide a proof for the correct preservation of the blocks $(m_{s_1}, \cdots, m_{s_c})$ of file $F = (m_1, \cdots, m_n)$ under *Chal* in time period j^*, where $1 \le s_l \le n$, $1 \le l \le c$, and $1 \le c \le n$.
5. Forgery phase. Adversary $\mathcal{A}$ outputs a proof P for the preservation of the blocks $(m_{s_1}, \cdots, m_{s_c})$ under *Chal* in time period j^*. $\mathcal{A}$ wins the game if ProofVerify(PK, j^*, *Chal*, P)="True".

Definition 4 (Forward Security(Key-Exposure Resilience) under Continual Leakage)**.** An auditing protocol is *l-Forward Secure(Key-Exposure Resilient) under Continual Leakage* if for any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage of the adversary to win the above game is a negligible function in λ.

4.2. Our protocol

Let Ψ be the auditing protocol that is secure under continual key-leakage, as described in Section 3. Let Ξ be the continual key-leakage resilient signature scheme used in Ψ. We construct a new binary-tree-based auditing protocol from Ψ by following an existing approach [20].

1) Notations and Structures: We first describe the structures of our protocol at a high level. The entire lifetime of data is divided into discrete time periods $0, \cdots, T-1$. Without loss of generality, we assume $T = 2^l$. We employ a binary tree structure to designate these time periods. The leaf nodes of the tree from the leftmost one to the rightmost one represent the independent auditing protocol Ψ that is used to audit the data integrity from time period 0 to $T-1$. That is, the depth of the binary tree is $l = log_2 T$. The depth of the root node is 0. The leftmost two leaf nodes representing time periods 0 and 1 are generated in the *system setup process*. Other leaf nodes are generated by the *key update algorithm* as follows: we generate two leaf nodes representing two time periods $j = 2t$ and $j+1$ $(t = 1 \sim \frac{T}{2} - 1)$ with the same parent node at a time when the time period changes from $j-1$ to j for the following time periods. A leaf node of the tree corresponding to time period j is erased when the time period changes from j to $j+1$, which is implemented by the key update procedure. As [20], each node of the tree contains an independently generated Ψ public key and the corresponding secret key, and the signature of the Ψ public key under the secret key of its parent node is also included, except for the root node. The secret key is erased immediately once both child nodes of the node have been generated, thereby ensuring the forward security of the constructed auditing protocol.

Each node is appointed to a binary string w to represent the path from the root node to the node. Let the binary string of the root node be $w = 1$. The binary string of a node can be constructed as follows: let w be the binary string representing its parent node. If it is the left child node, its representing binary string is $w||0$; otherwise it is $w||1$. This construction indicates that the maximum length of w is $l + 1$. We provide an example of a binary tree with a depth of 2 for multiple time periods in Fig. 6.

An item of Tree[], i.e. structure of a node:

PK^w	Public key of this node
σ	Signature of PK^w under its parent's secret key
ts	Time period
SK^w	Secret key of this node
w	Binary string representing the node

Auth[]: l+n spaces

(PK_1, σ_1)	Public key of the first node in the path and the signature under root node's secret key
(PK_2, σ_2)	Public key of the second node in the path and the signature under the first node's secret key
……	……
(PK_l, σ_l)	Public key of the leaf node in the path and the signature under its parent node's secret key
ϕ	The authenticator generated by ψ

P[]: l+2 spaces

(PK_1, σ_1)	Public key of the first node in the path and the signature under root node's secret key
(PK_2, σ_2)	Public key of the second node in the path and the signature under the first node's secret key
……	……
(PK_l, σ_l)	Public key of the leaf node in the path and the signature under its parent node's secret key
μ	The linear combination of the challenged blocks
ρ	The aggregated authenticator generated by ψ

Fig. 7. Structures.

We use an array named *Tree*[] to store the entire binary tree with 2^{l+1} storage space. *Tree*[w] stores the node of which the representing binary string is w, which is a tuple as (PK^w, σ^w, ts^w, SK^w, w). Here, PK^w is the independently generated Ψ public key, and SK^w is the corresponding secret key. Further, σ^w is the signature of PK^w signed by Ξ under the secret key of its parent node, and ts^w is the time period.

The public key of the auditing protocol is the Ψ public key of the root node which is constant during the entire lifetime. The secret key $SK_{j,*}$ corresponding to time period j consists of the current state of the entire binary tree and the representing string of the leaf node corresponding to the current time period.

The authenticator of a message m includes an authenticator of m generated by Ψ.AuthGen under the secret key of the current leaf node, along with a chain of pairs(*public key,signature of public key under the node's parent's secret key*) of each non-root node on the path from the root node to this leaf. This ensures that the authenticator can be verified with only the public key of the root node. We use $\Phi = (ts, Auth[])$ to store the authenticator of F, where ts is the time period and *Auth*[] is an array in which to store the chain and the authenticator generated by Ψ. As the secret key used to generate authenticator is in a leaf node, the number of non-root nodes on the path from the root node to this leaf is l. Let $F = \{m_1, \cdots, m_n\}$, then *Auth*[] needs $l + n$ spaces. We also use an array P with $l + 2$ spaces in which to store the proof. The first l spaces store the above chain of pairs(*public key,signature of public key under the node's parent's secret key*). The last two spaces are used to store the linear combination of m_i, i.e., $\mu = \Sigma_{i\in I} v_i m_i$, and the aggregated authenticator generated by Ψ.AuthGen. We provide an example of the structure of the secret key, the authenticator, and the proof of our forward-secure auditing protocol in Fig. 7.

Algorithm 1 syssetup.

Input: input λ, T
Output: output public key and secret key

Set an array $Tree[0 \cdots 2T-1] = \{(\perp, \perp, \perp, \perp, \perp)\}$; $w =$"1"; $l = log_2 T$;
$(PK, SK) \leftarrow \Psi.\text{SysSetup}(1^\lambda)$;
$Tree[\text{StrToInt}(w)]=(PK, \perp, 0, SK, w)$;
$TmpSk = SK$;
while $|\ w\ | < l+1$ **do**
  $(PK_1, SK_1) \leftarrow \Psi.\text{SysSetup}(1^\lambda)$;
  $(PK_2, SK_2) \leftarrow \Psi.\text{SysSetup}(1^\lambda)$;
  $Tree[\text{StrToInt}(w \parallel 0)]=$
    $(PK_1, \Xi.\text{Sig}(PK_1, TmpSk), 0, SK_1, w \parallel 0)$;
  $Tree[\text{StrToInt}(w \parallel 1)]=$
    $(PK_2, \Xi.\text{Sig}(PK_2, TmpSk), 0, SK_2, w \parallel 1)$;
  EraseSk($Tree[\text{StrToInt}(w)]$);
  $TmpSk = SK_1$; $w = w \parallel 0$;
return PK, $(Tree, w)$;

Algorithm 2 keyup.

Input: input $SK_{j,i-1}$, j
Output: output the initial secret key $SK_{j+1,0}$ in time period $j+1$

Parse $SK_{j,i-1}$ as $(Tree, w^j)$; $w = w^j$;
if $j == \frac{T}{2} - 1$ **then**
  while $|\ w\ | > l$ **do**
    EraseNode($Tree, w$);
    $w = w[0, |\ w\ | -1]$;
else
  $w = w[0, |\ w\ | -1]$;
  if $Tree[\text{StrToInt}(w \parallel 0)] \neq \perp$ **then**
    EraseNode($Tree, w \parallel 0$);
    $Tree[\text{StrToInt}(w \parallel 1)].\text{ts}=j+1$;
    return $(Tree, w \parallel 1)$;
  while $Tree[\text{StrToInt}(w \parallel 0)]==\perp$ **do**
    EraseNode($Tree, w \parallel 1$);
    $w = w[0, |\ w\ | -1]$;
  EraseNode($Tree, w \parallel 0$);
$w = w \parallel 1$;
$TmpSk = Tree[\text{StrToInt}(w)].SK^w$;
while $|\ w\ | < l+1$ **do**
  $(PK_1, SK_1) \leftarrow \Psi.\text{SysSetup}(1^\lambda)$;
  $(PK_2, SK_2) \leftarrow \Psi.\text{SysSetup}(1^\lambda)$;
  $Tree[\text{StrToInt}(w \parallel 0)]=$
    $(PK_1, \Xi.\text{Sig}(PK_1, TmpSk)$,j+1,$SK_1, w \parallel 0)$;
  $Tree[\text{StrToInt}(w \parallel 1)]=$
    $(PK_2, \Xi.\text{Sig}(PK_2, TmpSk)$,j+1,$SK_2, w \parallel 1)$;
  EraseSk($Tree[\text{StrToInt}(w)]$);
  $TmpSk = SK_1$; $w = w \parallel 0$;
$Tree[\text{StrToInt}(w)].\text{ts}=j+1$;
return $(Tree, w)$;

2) Description of Our Protocol: Let w^j be the string representing the leaf node corresponding to time period j, and $|w|$ be the length of w. Let $w[q, h]$ denote the substring $w[q] \cdots w[h]$.

- SysSetup(1^λ, T) $\rightarrow$ (PK, $SK_{0,0}$): Let λ be the security parameter and T be the total number of time periods. It runs (PK, $SK_{0,0}$) $\leftarrow$ **syssetup**(1^λ, T) described in Algorithm 1.
- KeyUpdate(PK, j, $SK_{j,i-1}$) $\rightarrow$ ($SK_{j+1,0}$): It runs $SK_{j+1,0} \leftarrow$**keyup**($SK_{j,i-1}$, j) described in Algorithm 2 to update the secret key.

- AuthGen($PK, j, SK_{j,i-1}, F$) $\rightarrow$ (Φ): Let the file $F = \{m_1, \cdots, m_n\}$ be identified by *name*, where $m_i \in Z_N (i = 1, \cdots, n)$. The client computes the authenticator $\Phi \leftarrow$ **authgen**($PK, SK_{j,i-1}, F$) described in Algorithm 3.

Algorithm 3 authgen.

Input: input $PK, SK_{j,i-1}, F$
Output: output the authenticator of F

Parse $SK_{j,i-1}$ as $(Tree, w^j)$; $w = w^j$;
Parse $F = \{m_1, \cdots, m_n\}$;
$Auth[0 \sim l + n - 1] = \{0\}$;
$Auth[l \sim l + n - 1]$=
    Ψ.AuthGen($PK, Tree$[StrToInt(w)].SK^w, F);
$k = 1$;
while $|\ w\ | > 1$ **do**
    $Auth[l - k]$=
        ($Tree$[StrToInt(w)].PK, $Tree$[StrToInt(w)].σ);
    $w = w[0, |\ w\ | - 1]$;
    $k = k + 1$;
return $\Phi = (j, Auth)$;

- ProofGen(*PK, j, Chal, F,* Φ) $\rightarrow$ (P): The client randomly selects and sends to the cloud a challenge $Chal = \{(i, v_i)\}_{i \in I}$, where $I = \{s_1, \cdots, s_c\}$ is a c-element subset of set $[1, n]$ and $v_i \in Z_N$. The cloud generates a proof $P \leftarrow$ **proofgen**(*PK, j, Chal, F,* Φ) described in Algorithm 4.

Algorithm 4 proofgen.

Input: input $PK, j, Chal, F, \Phi$
Output: output a proof P

Parse $Chal = \{(i, v_i)\}_{i \in I}$;
Parse $F = \{m_1, \cdots, m_n\}$;
Parse $\Phi = (j, Auth)$;
if $j \neq \Phi.j$ **then**
    return error;
$\mu = \sum_{i \in I} v_i m_i$;
$P[0 \sim l - 1] = Auth[0 \sim l - 1]$;
$P[l] = \Psi$.ProofGen($PK, Chal, F, \Phi$);
$P[l + 1] = \mu$;
return P;

- ProofVerify(*PK, j, Chal, P*) $\rightarrow$ ("True" or "False"): By receiving the proof P, the client can verify the proof by executing **proofver**(*PK, j, Chal, P*) described in Algorithm 5.

Algorithm 5 proofver.

Input: input $PK, j, Chal, P$
Output: output "True" or "False"

Parse $Chal = \{(i, v_i)\}_{i \in I}$;
$TmpPk = PK$; $i = 0$;
for $i < l$ **do**
    if Ξ.Ver($TmpPk, P[i].PK, P[i].\sigma$)=="False" **then**
        return error;
    $TmpPk = P[i].PK$;
    $i = i + 1$;
return Ψ.ProofVerify($TmpPk, Chal, P[l \sim l + 1]$);

In these algorithms, StrToInt is a function that transforms a binary string into its value, EraseSk is used to delete the secret key associated with a tree node and EraseNode is a function to delete a node from the tree. We can prove the security by applying the same technique used in [20]. The continual key-leakage resilience of Ψ ensures the forward security(key-exposure resilience) under continual leakage of the above auditing protocol. We omit it here.

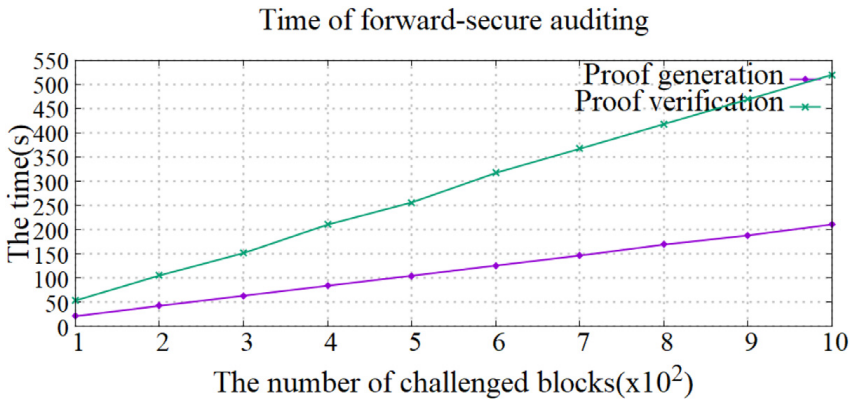


Fig. 8. The time of auditing procedures with different number of checked blocks.

4.3. Performance analysis

We use the same parameters as in the protocol in Section 3, i.e., the order of the curve group has $N = 192 \times 4 = 768$ bits, and the same testing environment. For simplification, we set the total time period $T = 16$ and the depth of the binary tree $l = 4$. The protocol is based on the continual key-leakage resilient protocol Ψ described in Section 3. The description of the protocol indicates that the size of the challenge message is the same as that in Ψ. The size of the proof message equals that in Ψ plus the size of l signatures signed by Ξ, which is the continual key-leakage signature in Section 2, i.e., $488 + 392 \times l$ bytes. The duration of the process to generate the authenticator is what in Ψ plus the time required for memory copying and *string-to-integer* converting operations. The duration of the process to generate the challenge is the same as that in Ψ, and the duration of the proof verification process is $l + 1$ times that in Ψ. The duration of the proof generation process is what in Ψ plus the time required for memory copying operations. Fig. 8 shows the duration of the proof generation and proof verification processes with different number of checked data blocks.

5. Conclusion

In this paper, we focus on providing a cloud auditing protocol with forward security under continual key-leakage. We feed a new security definition named "key-exposure resilience under continual leakage" to the auditing protocol and initiate the first attempt to construct an auditing protocol with this definition of security. This protocol enables the integrity of the data uploaded to the cloud to be successfully verified during the time period before that in which the client's current key exposure occurred even if the client's secret keys were partially leaked during previous periods. To this end, we first defined the formal security model of the auditing protocol with continual key-leakage resilience, and proposed the first concrete protocol. Then, we used an existing technique [20] to extend this protocol such that it provides key-exposure resilience under continual key-leakage.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Chengyu Hu: Conceptualization, Methodology, Software, Investigation, Writing - original draft, Writing - review & editing. **Yuqin Xu:** Software, Investigation. **Pengtao Liu:** Investigation, Writing - original draft. **Jia Yu:** Methodology, Writing - original draft. **Shanqing Guo:** Methodology, Writing - review & editing. **Minghao Zhao:** Software, Writing - original draft.

Acknowledgments

This project is supported in part by National Natural Science Foundation of China (no.61602275, 61632020, 61772311), Major Scientific and Technological Innovation Projects of Shandong Province, China (no.2019JZZY010132), Shandong Province Higher Educational Science and Technology Program (no.J15LN01), the Open Project of Key Laboratory of Network Assessment Technology, Institute of information engineering, Chinese Academy of Sciences (no.KFKT2019-002), the Open Project of Co-Innovation Center for Information Supply & Assurance Technology, Anhui University (no.ADXXBZ201702).

References

[1] Markets, Markets, Cloud Storage Market by Type, Deployment Model, Organization Size, Vertical, and Region - Global Forecast to 2022, 2018, https://www.marketsandmarkets.com/Market-Reports/cloud-storage-market-902.html.
[2] N. Bonvin, T.G. Papaioannou, K. Aberer, A Self-organized, Fault-tolerant and Scalable Peplication Scheme for Cloud Storage, in: Proceedings of the 1st ACM Symposium on Cloud Computing (SoCC), ACM, 2010, pp. 205–216.
[3] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin, et al., Erasure Coding in Windows Azure Storage., in: Proceedings of the Usenix Annual Technical Conference (ATC), Boston, MA, 2012, pp. 15–26.
[4] Z. Whittaker, Amazon web services suffers partial outage, 2012,
[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable Data Possession at Untrusted Stores, in: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), ACM, 2007, pp. 598–609.
[6] C.C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, ACM Trans. Inf. Syst. Secur. (TISSEC) 17 (4) (2015) 15.
[7] P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, in: Proceedings of the Annual International Cryptology Conference (CRYPTO), Springer, 1999, pp. 388–397.
[8] E. Brier, C. Clavier, F. Olivier, Correlation power analysis with a leakage model, in: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer, 2004, pp. 16–29.
[9] P.C. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, in: Proceedings of the Annual International Cryptology Conference (CRYPO), Springer, 1996, pp. 104–113.
[10] K. Gandolfi, C. Mourtel, F. Olivier, Electromagnetic analysis: concrete results, in: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer, 2001, pp. 251–261.
[11] S. Jana, V. Shmatikov, Memento: Learning Secrets from Process Footprints, in: Proceedings of the 2012 IEEE Symposium on Security and Privacy (S&P), IEEE, 2012, pp. 143–157.
[12] M. Vuagnoux, S. Pasini, Compromising electromagnetic emanations of wired and wireless keyboards., in: Proceedings of the USENIX Security Symposium, 2009, pp. 1–16.
[13] R. Raguram, A.M. White, D. Goswami, F. Monrose, J.-M. Frahm, iSpy: Automatic Reconstruction of Typed Input from Compromising Reflections, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, ACM, 2011, pp. 527–536.
[14] D. Genkin, I. Pipman, E. Tromer, Get Your Hands Off My Laptop: Physical Side-Channel Key-Extraction Attacks on PCs, in: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer, 2014, pp. 242–260.
[15] J. Yu, K. Ren, C. Wang, V. Varadharajan, Enabling cloud storage auditing with key-exposure resistance, IEEE Trans. Inf. Forensics Secur. 10 (6) (2015) 1167–1179.
[16] J. Yu, H. Wang, Strong key-exposure resilient auditing for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 12 (8) (2017) 1931–1940.
[17] X. Zhang, H. Wang, C. Xu, Identity-based key-exposure resilient cloud storage public auditing scheme from lattices, Inf. Sci. 472 (2019) 223–234.
[18] R.P. McEvoy, C.C. Murphy, W.P. Marnane, M. Tunstall, Isolated wddl: a hiding countermeasure for differential power analysis on fpgas, ACM Trans. Reconfigurable Technol. Syst. (TRETS) 2 (1) (2009) 3.
[19] J.D. Golić, C. Tymen, Multiplicative masking and power analysis of aes, in: International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer, 2002, pp. 198–212.
[20] M. Bellare, A. O'Neill, I. Stepanovs, Forward-security under continual leakage, in: Proceedings of the 16th International Conference on Cryptology and Network Security (CANS), 2017, pp. 3–26.
[21] J. Yu, R. Hao, H. Xia, H. Zhang, X. Cheng, F. Kong, Intrusion-resilient identity-based signatures: concrete scheme in the standard model and generic construction, Inf. Sci. 442–443 (2018) 158–172.
[22] M. Naor, G.N. Rothblum, The Complexity of Online Memory Checking, in: Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, 2005, pp. 573–582.
[23] A. Juels, B.S. Kaliski Jr, PORs: Proofs of Retrievability for Large Files, in: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), Acm, 2007, pp. 584–597.
[24] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, D. Song, Remote data checking using provable data possession, ACM Trans. Inf. Syst. Secur. (TISSEC) 14 (1) (2011) 12.
[25] E. Shi, E. Stefanov, C. Papamanthou, Practical dynamic proofs of retrievability, in: Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security, ACM, 2013, pp. 325–336.
[26] Z. Ren, L. Wang, Q. Wang, M. Xu, Dynamic proofs of retrievability for coded cloud storage systems, IEEE Trans. Serv. Comput. 11 (4) (2018) 685–698.
[27] C. Wang, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for data storage security in cloud computing, in: Proceedings of the International Conference on Computer Communications (INFOCOM), IEEE, 2010, pp. 1–9.
[28] F. Armknecht, J.-M. Bohli, G.O. Karame, Z. Liu, C.A. Reuter, Outsourced proofs of retrievability, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 831–843.
[29] H. Tian, Y. Chen, C.-C. Chang, H. Jiang, Y. Huang, Y. Chen, J. Liu, Dynamic-hash-table based public auditing for secure cloud storage, IEEE Trans. Serv. Comput. 10 (5) (2017) 701–714.
[30] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 14 (2) (2019) 331–346.
[31] C. Wang, S.S. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, IEEE Trans. Comput. 62 (2) (2013) 362–375.
[32] Y. Yu, M.H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G. Min, Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage, IEEE Trans. Inf. Forensics Secur. 12 (4) (2017) 767–778.
[33] M. Etemad, A. Küpçü, Transparent, distributed, and replicated dynamic provable data possession, in: International Conference on Applied Cryptography and Network Security (ACNS), Springer, 2013, pp. 1–18.
[34] H.C. Chen, P.P. Lee, Enabling data integrity protection in regenerating-coding-based cloud storage: theory and implementation, IEEE Trans. Parallel Distrib. Syst. 25 (2) (2014) 407–416.
[35] J. Liu, K. Huang, H. Rong, H. Wang, M. Xian, Privacy-preserving public auditing for regenerating-code-based cloud storage, IEEE Trans. Inf. Forensics Secur. 10 (7) (2015) 1513–1528.
[36] S. Goldwasser, Multi party computations: past and present, in: Proceedings of the sixteenth annual ACM Symposium on Principles of Distributed Computing (PODC), ACM, 1997, pp. 1–6.
[37] A.C. Yao, Protocols for secure computations, in: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS), IEEE, 1982, pp. 160–164.
[38] E. Boyle, S. Goldwasser, A. Jain, Y.T. Kalai, Multiparty Computation Secure Against Continual Memory Leakage, in: Proceedings of the 44h Annual ACM Symposium on Theory of Computing (STOC), ACM, 2012, pp. 1235–1254.
[39] E. Boyle, S. Garg, A. Jain, Y.T. Kalai, A. Sahai, Secure Computation Against Adaptive Auxiliary Information, in: Advances in Cryptology–CRYPTO 2013, Springer, 2013, pp. 316–334.
[40] N. Bitansky, D. Dachman-Soled, H. Lin, Leakage-tolerant Computation with Input-independent Preprocessing, in: Proceedings of the International Cryptology Conference (CRYPTO), Springer, 2014, pp. 146–163.
[41] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[42] V. Attasena, J. Darmont, N. Harbi, Secret sharing for cloud data security: a survey, VLDB J. –Int. J. Very Large Data Bases (VLDBJ) 26 (5) (2017) 657–681.

[43] F. Benhamouda, A. Degwekar, Y. Ishai, T. Rabin, On the local leakage resilience of linear secret sharing schemes, in: Annual International Cryptology Conference, Springer, 2018, pp. 531–561.
[44] A. Srinivasan, P.N. Vasudevan, Leakage resilient secret sharing and applications(2018). https://eprint.iacr.org/2018/1154.pdf.
[45] D. Aggarwal, I. Damgard, J.B. Nielsen, M. Obremski, E. Purwanto, J. ao Ribeiro, M. Simkin, Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures, IACR eprint (2018). https://eprint.iacr.org/2018/1147.pdf
[46] C. Hu, Z. Li, P. Liu, R. Yang, S. Guo, H. Zhang, Verifiable public-key encryption with keyword search secure against continual memory attacks, Mobile Networks and Applications (2018) 1–11.
[47] S. Dai, H. Li, F. Zhang, Memory leakage-resilient searchable symmetric encryption, Future Generation Computer Systems 62 (2016) 76–84.
[48] X. Ge, J. Yu, H. Zhang, C. Hu, Z. Li, Z. Qin, R. Hao, Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification, IEEE Transactions on Dependable and Secure Computing (2019). 10.1109/TDSC.2019.2896258
[49] A. Lewko, M. Lewko, B. Waters, How to leak on key updates, in: Proceedings of the 43rd ACM Symposium on Theory of Computing, (STOC 2011), ACM, 2011, pp. 725–734.
[50] A. Ateniese, R. Burns, R. Curtmola, J. Herring, Provable data possession at untrusted stores, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, (CCS 2007), ACM, 2007, pp. 598–609.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON CLOUD - FOG - EDGE COMPUTING IN CYBER-PHYSICAL-SOCIAL SYSTEMS (CPSS)

Received January 1, 2020, accepted January 13, 2020, date of publication January 17, 2020, date of current version January 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967457

Public-Key Encryption Secure Against Related Randomness Attacks for Improved End-to-End Security of Cloud/Edge Computing

PENGTAO LIU
College of Cybersecurity, Shandong University of Political Science and Law, Jinan 250014, China
e-mail: ptwave@163.com

This work was supported in part by the Shandong Province Higher Educational Science and Technology Program under Grant J15LN01.

ABSTRACT Public-key encryption is often used to protect data security/privacy and secure communication in scenarios of cloud computing and edge computing. Related randomness attacks model (RRA) for public-key encryption was motivated by randomness failures. This paper proposes some methods of constructing secure public-key encryption scheme against related randomness attacks, i.e. RRA-CPA secure public-key encryption scheme with efficient decryption algorithm and short ciphertexts size obtained from one-way function with weak RKA-security and indistinguishability obfuscation, RRA-CPA secure public-key encryption scheme against arbitrarily function from any publicly deniable encryption and RRA-CCA secure public-key encryption scheme against arbitrarily function from standard IND-CCA public-key encryption scheme with a hardcore function for arbitrarily correlated inputs.

INDEX TERMS Arbitrary restricted function, publicly deniable encryption, public-key encryption, related randomness attack.

I. INTRODUCTION

The rise of Internet, Internet of Things and Cloud Computing, as well as the rapid popularization of mobile devices, intelligent terminals, social networks and e-commerce, has led to the rapid growth of data volume. Cloud computing provides a technical platform and often recommends services to users by cloud service recommendation [1]–[3] for the storage, computation and management of large data, which makes the processing of large data more convenient and efficient. As cloud computing releases the user's burden in maintaining basic storage infrastructures, many individuals and institutions adopt cloud storage to maintain their data so that more and more data are stored on cloud servers. However, cloud servers may tamper with, delete and damage data. Although there are many techniques to improve cloud security [4], [5], the simplest method for the data owners to share the data while keeping the security and privacy is encrypting data using public-key encryption and uploading it to cloud servers.

Traditional cloud computing cannot handle the huge data generated by network edge devices which makes the birth of edge computing solving the problem in combination with cloud computing [6], [7] which scenario is showed in Fig. 1. Edge computing is widely used in the Internet of things, especially in the application scenarios with special requirements such as low delay, high bandwidth, high reliability, massive connections, heterogeneous convergence, local security and privacy-preserving [8]–[15]. The edge computing model requires secure communication among the clouds and the edges. Public-key encryption (PKE) is suitable for the end-to-end security of edge computing participants and is

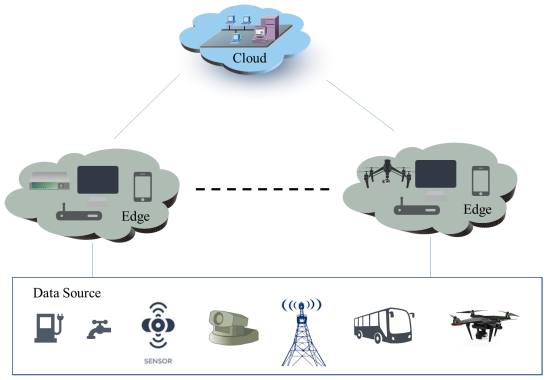


FIGURE 1. Cloud-edge computing scenario.

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaokang Wang.

often used for exchanging a session key between unknown partners according to its advantages in secure broadcast and authentication, while there are many techniques to ensure the edge computing security [16], [17].

The development of information technology has also led to further integration of information, physical systems and human society to form a more complex system, i.e. Cyber-physical-social system (CPSS). The strong coupling of CPSS brings security and privacy issues, e.g. multi-source spatial data can be associated to leak user privacy. Many technologies can be applied to protect the data privacy of cyber-physical-social system, including anonymity, trusted computing, encryption, verifiable computing and data obfuscation, in which public-key encryption can be used for security and access control in cyber-physical-social systems.

Randomness quality is crucial to the security of cryptosystems that consuming lots of randomnesses. However, many researches such as [18]–[21] show the randomness failures including randomness reuse, randomness tampering, etc. For cloud/edge computing scenario, [22] showed that the securely kept DSA keys used in TLS authentication sessions could be extracted by virtual-machine reset attacks due to the randomness repetition. These failures can cause some security problems such as signing keys exposure, plaintext recovery, weak key generation and so on. Motivated by preserving security under randomness failures, related randomness attacks model (RRA) for PKE was abstracted and secure cryptosystems were constructed in this model. Informally, in this model, the adversary has the ability to control the randomness used during encryption and the indistinguishability of ciphertexts should be kept on this condition.

In this paper, we focus on constructing secure PKE schemes in the model of related randomness attacks. First, we propose a method of constructing RRA secure PKE scheme *under chosen plaintext attack* (RRA-CPA) based on indistinguishability obfuscation and one-way function with weak RKA-security. To acquire RRA security against arbitrary function, we give two constructions. The first one is publicly deniable encryption which can be proved to have RRA security *under chosen plaintext attack* (RRA-CPA), and the other one is constructed by combining a secure standard PKE scheme *under chosen ciphertext attack* (IND-CCA) with a hardcore function for arbitrarily correlated inputs which can achieve RRA security *under chosen ciphertext attack*.

A. RELATED WORKS

Recently, many works address the problem of randomness failures. Austrin *et al.* [23] showed a negative result that standard encryption schemes can be broken under randomness failure. Feltz and Cremers [24] analyzed the authenticated key exchange protocols and showed that bad randomness results in the insecurity of the protocols. Paterson *et al.* [25] introduced a security model for PKE schemes called "*related randomness attacks model*", in which the adversary has the ability to force the usages of related randomness in encryption which are abstracted to the outputs of specified functions applied to some initial randomness. They also proposed many kinds of schemes in this model. They showed that a RRA-secure PKE in the random oracle model can be obtained by taking the hash value of the input random together with the message and public key as the rand coins used in encryption. To construct a RRA-secure PKE scheme in the standard model, they applied pseudorandom function secure against related-key attacks (RKA-PRF) to a standard PKE scheme. The restricted function families depend on that of RKA-PRF which currently are restricted to the function families consisting of polynomials of bounded degree according to the instantiations of available RKA-PRFs. To obtain further constructions for other kinds of restricted function families, Paterson et al. considered weakened security models; they first restricted honest generations of the public keys (HK-RRA) and provided a generic method of constructing RRA-secure schemes in this condition by taking the value hashed by a Correlated-Input Secure (CIS) hash function proposed in [26] instead of a standard hash function as the randomness used in encryption process of a PKE scheme; then they considered the situation with no restriction on public keys where the adversary is restricted to use a vector of prefixed functions to implement its attack (FV-RRA) and gave a concrete FV-RRA secure scheme under chosen-plaintext attack under the DDH assumption where the used functions are hard-to-invert ones. To achieve FV-RRA security under chosen-ciphertext attack, Paterson *et al.* [27], presented a general transformation for PKE using a value extracted by an auxiliary input reconstructive extractor as randomness. Yuen *et al.* [28] proposed some related randomness attack models which cover related key/randomness attacks for PKE and digital signature, and provided generic constructions for security against these attacks. Schuldt and Shinagawa [29] analyzed the related-randomness security about RSA-OAEP and gave a positive result. A basic comparison between our schemes and two typical RRA secure schemes is shown in Table I.

TABLE 1. Comparison of schemes.

Scheme	CCA/ CPA	Arbitrary restricted function	Efficient Decryption
[25]	CCA	×	×
[28]	CCA	×	×
Our scheme I	CPA	√	√
Our scheme II	CPA	√	×
Our scheme III	CCA	√	×

B. OUR CONTRIBUTIONS

In consideration of the importance of resisting related randomness attacks in PKE scheme, we focus on how to build secure PKE schemes against related randomness attacks in this paper. Our contributions are summarized in the following:

(1) Propose a method of constructing RRA-CPA secure PKE scheme from indistinguishability obfuscation and weak RKA-secure one-way function. We first give a weak

definition of RKA-secure one-way function (wRKAOWF) and show how to construct weak RKA-secure one-way function. Then, we build related-seed secure pseudorandom generator using weak RKA-secure one-way function and a hardcore function for arbitrarily correlated inputs proposed in [30]. At last, we replace the standard pseudorandom generator used in the PKE scheme in [31] with a related-seed secure pseudorandom generator and prove the RRA-CPA security of the new scheme. The construction is simple, and has advantage in the efficiency in the decryption algorithm and ciphertexts size.

(2) Construct RRA-secure PKE schemes against arbitrary function.

- Prove that any publicly deniable encryption with *Indistinguishability under Chosen Plaintext Attack(IND-CPA)* and *Indistinguishability of Explanation* [31] is RRA-CPA secure PKE against arbitrary function.
- Propose a method of constructing a RRA-CCA secure PKE scheme by applying a hardcore function for arbitrarily correlated inputs proposed in [30] to a standard IND-CCA secure PKE scheme. To encrypt messages, the construction first apply the hardcore function to a randomness r to obtain the output r'. Then it implements the encryption algorithm of PKE scheme taking r' as the actual random coin for encryption. At last, we prove the RRA-CCA security of the construction.

C. ORGANIZATION

Section 2 reviews some preliminaries. Section 3 gives a method of constructing RRA-CPA secure PKE scheme from weak RKA-secure OWF. We define weak RKA-security of one-way function and a concrete instance of it. We construct a RRA-CPA secure PKE scheme from indistinguishability obfuscation and weak RKA-secure one-way function. In Section 4, we describe in detail how to construct PKE schemes with RRA-security against arbitrary function. We prove that any publicly deniable encryption is a RRA-CPA secure PKE against arbitrary function. To obtain RRA-CCA secure secure PKE against arbitrary function, we combine the standard IND-CCA PKE scheme with hardcore function for arbitrarily correlated inputs. Finally, conclusions are drawn in Section 5.

II. PRELIMINARIES

A. PUBLIC-KEY ENCRYPTION

Let PKE = (Keygen, Encrypt, Decrypt) be a PKE scheme (PKE) [32]. Keygen is used to generate a public/secret key pair (pk, sk) randomly. The probabilistic algorithm Encrypt uses pk and a randomly chosen coin r from randomness space **Rnd** to encrypt a message $m \in \mathcal{M}$ to its corresponding cipher text c. Using a private key sk, the deterministic algorithm Decrypt can return the corresponding plain text m or an error symbol $\perp$ by decrypting a cipher text c.

There are two classical security definitions for PKE, i.e., *Indistinguishability under Chosen Plaintext Attack*(*IND-CPA* in abbreviation) and *Indistinguishability under Chosen Ciphertext Attack*, (*IND-CCA* in abbreviation). Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary of a PKE scheme Π, and the advantage of $\mathcal{A}$ to break the security of Π is defined as follows:

$$\mathrm{Adv}^{IND-ATK}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l| l} & (sk, pk) \leftarrow KeyGen(1^\lambda); \\ & (state, m_0, m_1) \\ b = b' : & \quad \leftarrow \mathcal{A}_1^{EO(pk,m),DO(sk,c)}(\cdot); \\ & b \leftarrow_R \{0,1\}; \\ & c^* \leftarrow LR(m_0, m_1); \\ & b' \leftarrow \mathcal{A}_2^{EO(pk,m),DO(sk,c)}(pk, c^*, state) \end{array}\right] - \frac{1}{2}$$

1) IND-ATK SECURITY (ATK=CPA,CCA)

A PKE scheme is called IND-ATK secure if for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in λ.

The *related randomness security under Chosen Plaintext/Ciphertext Attack*(*RRA-CPA/RRA-CCA* for short) is two formal security definition introduced by [25] for PKE. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary of a PKE scheme Π with a class of Φ of functions, and the advantage of $\mathcal{A}$ to break the related randomness security of Π is defined as follows:

$$\mathrm{Adv}^{\Phi-RRA-ATK}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l| l} & (sk, pk) \leftarrow KeyGen(1^\lambda); \\ & CoinTable \leftarrow \emptyset; \\ & (state, m_0, m_1) \\ b = b' : & \quad \leftarrow \mathcal{A}_1^{EO(pk,m,\phi),DO(sk,c)}(\cdot); \\ & b \leftarrow_R \{0,1\}; \\ & c^* \leftarrow LR(m_0, m_1, i, \phi); \\ & b' \leftarrow \mathcal{A}_2^{EO(pk,m,\phi),DO(sk,c)}(pk, c^*, state) \end{array}\right] - \frac{1}{2}$$

EO(pk, m, ϕ) and LR(m_0, m_1, i, ϕ) are oracles proceed as follows in Table II:

TABLE 2. Encrypt and LR oracles in RRA-ATK security game.

EO(pk, m, ϕ):	LR(m_0, m_1, i, ϕ):
If CoinTable$[i]=\perp$	If CoinTable$[i]=\perp$
CoinTable$[i] \leftarrow_R$**Rnd**	CoinTable$[i] \leftarrow_R$**Rnd**
$r_i \leftarrow$CoinTable$[i]$	$r_i \leftarrow$CoinTable$[i]$
$c \leftarrow$PKE.Encrypt$(pk, m, \phi(r_i))$	$c^* \leftarrow$PKE.Encrypt$(pk, m_b, \phi(r_i))$
Return c	Return c^*

2) RRA-ATK SECURITY (ATK=CPA,CCA)

A PKE scheme is called RRA-ATK secure if for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ is negligible in λ.

Note that *EO* is encrypt oracle and *DO* is decrypt oracle. The adversary's access to decrypt oracle *DO* is removed if ATK = CPA.

B. ONE-WAY FUNCTION

We use the description about one-way function in [33]. A function family **F** is *one-way* or called one-way function **OWF**, if the following advantage $\mathrm{Adv}^{ow}_{\mathbf{F},\mathcal{A}}(\lambda)$ for any probabilistic adversary $\mathcal{A}$ to break the one-wayness of **F** is a negligible function in λ.

$$\mathrm{Adv}^{ow}_{\mathbf{F},\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l| l} & k \leftarrow Key_F(1^\lambda) \\ x' & y \leftarrow Eval_F(k,x) \\ & x' \leftarrow \mathcal{A}(1^\lambda, k, y) \\ & Eval_F(k,x') = y \end{array}\right]$$

C. INDISTINGUISHABILITY OBFUSCATION

Let $\{\mathcal{C}_\lambda\}$ be a circuit class. It is called an indistinguishability obfuscator for $\{\mathcal{C}_\lambda\}$ if an uniform PPT algorithm $i\mathcal{O}$ satisfies the following conditions:

- For all $\lambda \in \mathbb{N}$, all $\mathcal{C} \in \{\mathcal{C}_\lambda\}$ and all input x
  $\Pr[\mathcal{C}'(x) = \mathcal{C}(x) : \mathcal{C}' \leftarrow i\mathcal{O}(\lambda, \mathcal{C})] = 1.$
- For any PPT algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, if $\Pr[\forall x, \mathcal{C}_0 = \mathcal{C}_1 : (\mathcal{C}_0, \mathcal{C}_1, \sigma) \leftarrow \mathcal{A}_1(1^\lambda)] > 1 - \epsilon(\lambda)$, then
  $|\Pr[\mathcal{A}_2(\sigma, i\mathcal{O}(\lambda, \mathcal{C}_0)) = 1 : (\mathcal{C}_0, \mathcal{C}_1, \sigma) \leftarrow \mathcal{A}_1(1^\lambda)] -$
  $\Pr[\mathcal{A}_2(\sigma, i\mathcal{O}(\lambda, \mathcal{C}_1)) = 1 : (\mathcal{C}_0, \mathcal{C}_1, \sigma) \leftarrow \mathcal{A}_1(1^\lambda)]| \leq \epsilon(\lambda)$
  where $\epsilon(\lambda)$ is a negligible function.

The existences of this kind of $i\mathcal{O}$ for all polynomial size circuits were given in [34].

D. PUNCTURABLE PSEUDORANDOM FUNCTION

A puncturable family of PRFs **F** described in [31] can be defined by three algorithms Key_F,$Puncture_F$, and $Eval_F$, and two computable functions $n(\cdot)$ and $m(\cdot)$, which satisfy the following conditions:

-Functionality preserved under puncturing. The following equality is established for every PPT adversary $\mathcal{A}$ and for all $x \in \{0,1\}^{n(\lambda)}$, $x \notin S$ where $S \subseteq \{0,1\}^{n(\lambda)}$ is output by adversary $\mathcal{A}$:

$$\Pr\left[\begin{array}{l} Eval_F(K,x) = : K \leftarrow Key_F(1^\lambda) \\ Eval_F(K_S, x) \quad K_S \leftarrow Puncture_F(K, S) \end{array}\right] = 1$$

-Pseudorandom at punctured points. For a negligible function $negl(\cdot)$ and every PPT adversary $(\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$ and state σ, then:

$$|\Pr[\mathcal{A}_2(\sigma, K_S, S, Eval_F(K,S)) = 1] - \Pr[\mathcal{A}_2(\sigma, K_S, S, U_{m(\lambda)\cdot|S|}) = 1]| = negl(\lambda)$$

where $S = (x_1, \cdots, x_k)$, $K \leftarrow Key_F(1^\lambda)$, $K_S \leftarrow Puncture_F(K,S)$ and $Eval_F(K,S)$ denotes the concatenation of $Eval_F(K,x_1), \cdots, Eval_F(K,x_k)$, U_l denotes the uniform distribution over l bits.

E. HARDCORE FUNCTION FOR CORRELATED INPUTS

In [30], the authors constructed hardcore functions to extract random bits on the condition where the inputs are arbitrarily correlated. Let **HC** be a family of functions and **F** be a one-way function family. **HC** is hardcore function for **F** if the following advantage $\mathrm{Adv}^{\mathcal{H}}_{\mathbf{F},\mathbf{HC},\mathcal{I}}$ of an adversary $\mathcal{H}$ is negligible in λ.

$$\mathrm{Adv}^{\mathcal{H}}_{\mathbf{F},\mathbf{HC},\mathcal{I}}(\lambda) = \Pr\left[\begin{array}{l| l} & b \leftarrow \{0,1\} \\ & k \leftarrow Key_F(1^\lambda) \\ & hp \leftarrow \mathbf{HC.Pg}(1^\lambda) \\ b = b' & \vec{y} \leftarrow Eval_F(k, \vec{x}) \\ & \text{if } b = 1 \text{ then } \vec{r} \leftarrow \mathbf{HC}(hp, \vec{x}) \\ & \text{else } \vec{r} \leftarrow \mathbf{Rnd}(\vec{x}, \mathbf{HC}.ol(1^\lambda)) \\ & b' \leftarrow \mathcal{H}(1^\lambda, k, hp, \vec{y}, \vec{r}) \end{array}\right] - \frac{1}{2}$$

Here, we denote $\vec{y} \leftarrow Eval_F(k, \vec{x})$ as a vector of $y_i \leftarrow Eval_F(k, x_i)$ for each $x_i \in \vec{x}$. In [30], the authors gave a construction of hardcore function **HC** for *injective* one way functions **F** which can extract hardcore bits for arbitrarily correlated inputs.

III. RRA-SECURE PKE SCHEME FROM RKA-SECURE OWF

In [33], the authors proposed the definition of RKA-secure one-way function (RKAOWF) and applied it to construct RKA-secure signature schemes. However, their definition of RKAOWF is somewhat strong and cannot be used to construct RRA-secure PKE schemes. In this section, we will give a weak definition of RKAOWF and construct RRA-secure PKE schemes via it.

Firstly, we describe the definition of RKA-secure one-way function in [33] in the following. Let **F** be a function family. A class of polynomial-time computable functions Φ for **F** that specifies for each $\lambda \in \mathbb{N}$, each $k \in Key_F(1^\lambda)$ and each $\phi \in \Phi$ is called the related-key deriving (RKD) function.

Definition 1 (RKAOWF): **F** is called Φ-RKA secure if for any PPT adversary $\mathcal{A}$ for **F**, the following advantage $\mathrm{Adv}^{\Phi-RKA}_{\mathbf{F},\mathcal{A}}(\lambda)$ is a negligible function in λ.

$$\mathrm{Adv}^{\Phi-RKA}_{\mathbf{F},\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l| l} & k \leftarrow Key_F(1^\lambda) \\ x' & y \leftarrow Eval_F(k,x) \\ & x' \leftarrow \mathcal{A}^{Eval}(1^\lambda, k, y) \\ & Eval_F(k,x') = y \end{array}\right]$$

Eval(ϕ) is an oracle proceeds as follows:

$$\begin{array}{l} \mathrm{Eval}(\phi): \\ x' \leftarrow \phi(1^\lambda, k, x) \\ y' \leftarrow Eval_F(k, x') \\ return y' \end{array}$$

In the following, we give a weak definition of RKA-secure one-way function (wRKAOWF) which will be used to construct RRA-secure PKE schemes. Informally, wRKAOWF definition is as same as RKAOWF definition except that we weaken the adversary's advantage and let it return x' which satisfies that $Eval_F(k,x') = Eval_F(k, \Phi(1^\lambda, k, \phi, x))$.

Definition 2 (wRKAOWF): We say that **F** is Φ-wRKA secure if for any PPT adversary $\mathcal{A}$ for **F**, the following

advantage $\mathrm{Adv}^{\Phi-wRKA}_{\mathbf{F},\mathcal{A}}(\lambda)$ is a negligible function in λ.

$$\mathrm{Adv}^{\Phi-wRKA}_{\mathbf{F},\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l} k \leftarrow Key_F(1^\lambda) \\ y \leftarrow Eval_F(k,x) \\ x' \ : \ x' \leftarrow \mathcal{A}^{Eval}(1^\lambda, fp, y) \\ \mathbf{Eval_F}(1^\lambda, k, x') = \\ \quad Eval_F(k, \phi(1^\lambda, k, x)) \end{array}\right]$$

A. CONSTRUCTION OF wRKAOWF

In [33], it is proved that a function family $\mathbf{F}$ is $\Phi-$RKA secure if $\mathbf{F}$ is $\Phi-$key-malleable and one-way, where Φ is a class of RKD functions and $\Phi-$key-malleable property is defined as follows:

$\Phi-$key-malleable property([33]). Let $\mathbf{F}$ be a function family and Φ be a class of RKD functions for $\mathbf{F}$. $\mathbf{F}$ is called $\Phi-$key-malleable if there is a polynomial-time algorithm T, such that $T(1^\lambda, k, \phi, Eval_F(k,x)) = Eval_F(k, \phi(1^\lambda, k, x))$ for all $\lambda \in \mathbb{N}$, all $k \in [Key_F(1^\lambda)]$, all $\phi \in \{0,1\}^*$ and all x.

In the following, we will prove that if $\mathbf{F}$ is Φ-RKA secure and Φ is *root samplable* then $\mathbf{F}$ is Φ-wRKA secure. Here, we define *root samplable* class of functions below:

Definition 3 (Root Samplable): A function ϕ is root samplable if there exists a polynomial-time algorithm **CompR** which can uniformly output an element from $\phi^{-1}(0)$. A class of RKD functions Φ is root samplable if for each $\phi \in \Phi$ and each constant c, $\phi' = \phi - c$ is root samplable.

Note that the classes of linear functions, affine functions and polynomial functions over Z_p are root samplable as for a d-degree polynomial function f over Z_p, Ben-Or's algorithm [35] can compute the root of $f(x)$ in Z_p using $O((\log p)d^{2+e})$ operations in Z_p [7].

Theorem 1: Let $\mathbf{F}$ be a function family and Φ be a class of RKD functions for $\mathbf{F}$. If $\mathbf{F}$ is Φ-RKA secure and Φ is *root samplable* then $\mathbf{F}$ is Φ-wRKA secure.

Proof: Let $\mathcal{A}$ be a $\Phi-$wRKA adversary of $\mathbf{F}$. We can construct a $\Phi-$RKA adversary $\mathcal{B}$ of $\mathbf{F}$. On input $(1^\lambda, k, y)$, adversary $\mathcal{B}$ runs $\mathcal{A}$ and responds to $\mathcal{A}$'s Eval query by sending the same query to its challenger and returning the corresponding result. When $\mathcal{A}$ stops and outputs a value x' satisfying that $Eval_F(k,x') = Eval_F(k, \phi(1^\lambda, k, x))$ for a RKD function ϕ, adversary $\mathcal{B}$ can return x'' satisfying that $Eval_F(k,x'') = y$ by sampling the root x'' of $\phi(1^\lambda, k, x) - x'$. Note: If the advantage of $\mathcal{A}$ is ϵ then $\mathcal{B}$'s advantage is $\epsilon/poly(1^\lambda)$. If we assume $\mathbf{F}$ is injective, i.e. $\mathbf{F}$ is a family of one-way permutations, then $\mathcal{B}$'s advantage is ϵ.

Instance: In [33], the authors give three instances of RKAOWF for different classes of RKD functions. The first one-way function (permutation) described in Table 3 is based on discrete exponentiation in a cyclic group of prime order p and is Φ-RKA secure for a class Φ of affine RKD functions. Therefore, it is also Φ-wRKA secure one-way function (permutation). Next, we can prove that this one-way function (permutation) is also Φ-wRKA secure for a class Φ of polynomial RKD functions with upper bound $d = d(\lambda)$ of the degrees of the polynomials in $\Phi \le q$ as long as the $q-$SDL assumption described in the following holds.

TABLE 3. The instance.

k	x	$Eval_F(k,x)$	$\phi(1^\lambda, k, x)$
$(<G>, g, p)$	$\in Z_p^*$	g^x	$a_i \in Z_p, \sum_{i=0}^{d} a_i x^i \bmod p$

***q*-Strong Discrete Logarithm (*q*-SDL) Problem** [33]. Let $\mathbb{G}$ be a cyclic group. Let $\mathcal{A}$ be an algorithm outputting previously unknown random value $x \in Z_p^*$ with advantage $\mathrm{Adv}^{SDL}_{\mathcal{A},q}$ given $(g, g^x, g^{x^2}, \cdots, g^{x^q}) \in \mathbb{G}^{q+1}$, where $g \in \mathbb{G}$ is a generator of group $\mathbb{G}$ and

$$\mathrm{Adv}^{SDL}_{\mathcal{A},q} = \Pr[\mathcal{A}(g, g^x, g^{x^2}, \cdots, g^{x^q}) = x]$$

Definition 4: The q-SDL assumption holds in $\mathbb{G}$ if for all PPT algorithm $\mathcal{A}$, the advantage $\mathrm{Adv}^{SDL}_{\mathcal{A},q}$ of $\mathcal{A}$ in solving the q-SDL problem in $\mathbb{G}$ is negligible.

Proof: Now, we continue the proof of Φ-wRKA security. Let $\mathcal{A}$ be a PPT adversary breaking Φ-RKA security with an advantage ϵ with respect to the class of non-zero polynomials over Z_p. An algorithm $\mathcal{B}$ that solves a given random instance of q-SDL problem with the same advantage ϵ by interacting with $\mathcal{A}$ can be built as follows.

Given a random instance $(g, g^x, g^{x^2}, \cdots, g^{x^q}) \in \mathbb{G}^{q+1}$ of the q-SDL problem in $\mathbb{G}$, where $x \in Z_p^*$ is a unknown random value, $\mathcal{B}$ can output x as follows. $\mathcal{B}$ invokes $\mathcal{A}$ and gives $y = g^x$ to $\mathcal{A}$. Then $\mathcal{B}$ responds to $\mathcal{A}$'s Eval(ϕ) query with $y' = Eval_F(k, \phi(1^\lambda, k, x)) = g^{\sum_{i=0}^{d} a_i x^i} = \prod_{i=0}^{d} (g^{x^i})^{a_i}$. Eventually, $\mathcal{B}$ returns the same output x' of $\mathcal{A}$.

Definition 5: Related-Seed Pseudorandom Generator. We say that a pseudorandom generator (PRG) is $\Phi-$related-seed secure if for all PPT distinguisher $\mathcal{D}$

$$|\Pr[\mathcal{D}(r) = 1] - \Pr[\mathcal{D}(PRG(\phi(seed)) = 1]| \le \mathrm{negl}(\lambda)$$

where Φ is a class of RKD functions and $\phi \in \Phi$.

Construction of Related-Seed Pseudorandom Generator. Let $\mathbf{F}$ be a one-way permutation family and Φ be a class of RKD functions for $\mathbf{F}$, let $\mathbf{HC}$ be a hard-core predicate of $\mathbf{F}$. PRG=($Eval_F(k, seed)$,$\mathbf{HC}(seed)$) is proved to be a pseudorandom generator. Now, we can prove the following theorem.

Theorem 2: If $\mathbf{HC}$ is hardcore function for arbitrarily correlated inputs and $\mathbf{F}$ is Φ-wRKA secure, then the above PRG is $\Phi-$related-seed secure.

Proof: Let $\varepsilon(\lambda)$ be a non-negligible function. Assume that there exists a probabilistic polynomial-time distinguisher $\mathcal{D}$ such that

$$\varepsilon(\lambda) = \Big| \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k, \phi(s)), \mathbf{HC}(\phi(s))) = 1] - \Pr_{r\in\{0,1\}^{n+1}}[\mathcal{D}(r) = 1] \Big|$$

Note that

$$\Pr_{r\in\{0,1\}^{\lambda+1}}[\mathcal{D}(r) = 1] = \Pr_{r\in\{0,1\}^{\lambda}, r'\in\{0,1\}}[\mathcal{D}(r, r') = 1]$$

$$= \Pr_{s\in\{0,1\}^\lambda, r'\in\{0,1\}}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), r') = 1]$$
$$= \frac{1}{2}\cdot \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \mathbf{HC}(\phi(s))) = 1] + \frac{1}{2}\cdot \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \overline{\mathbf{HC}(\phi(s))}) = 1]$$

Therefore,

$$\varepsilon(\lambda) = \frac{1}{2}\cdot\Big|\Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \mathbf{HC}(\phi(s))) = 1] - \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \overline{\mathbf{HC}(\phi(s))}) = 1]\Big|.$$

Then an algorithm $\mathcal{A}$ can be constructed based on $\mathcal{D}$ to guess $\mathbf{HC}(\phi(s))$ given $y = \mathbf{Eval_F}(k,\phi(s))$. Upon inputting $y = \mathbf{Eval_F}(k,\phi(s))$ for a random s, algorithm $\mathcal{A}$ works as follows:

1) Choose $r' \in \{0,1\}$ uniformly.
2) Invoke $\mathcal{D}(y, r')$. If $\mathcal{D}$ returns 1, then output r'. Otherwise, output $\overline{r'}$.

We analyze the success probability of $\mathcal{A}$.

$$\begin{aligned}
&\Pr_{s\in\{0,1\}^\lambda}[\mathcal{A}(\mathbf{Eval_F}(k,\phi(s))) = \mathbf{HC}(\phi(s))]\\
&= \frac{1}{2}\cdot \Pr_{s\in\{0,1\}^\lambda}[\mathcal{A}(\mathbf{Eval_F}(k,\phi(s))) = \mathbf{HC}(\phi(s)) \mid r' = \mathbf{HC}(\phi(s))]\\
&\quad + \frac{1}{2}\cdot \Pr_{s\in\{0,1\}^\lambda}[\mathcal{A}(\mathbf{Eval_F}(k,\phi(s))) = \mathbf{HC}(\phi(s)) \mid r' \neq \mathbf{HC}(\phi(s))]\\
&= \frac{1}{2}\cdot\Big|\Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \mathbf{HC}(\phi(s))) = 1]\\
&\quad + \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \overline{\mathbf{HC}(\phi(s))}) = 0]\Big|\\
&= \frac{1}{2}\cdot\Big|\Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \mathbf{HC}(\phi(s))) = 1]\\
&\quad + (1 - \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \overline{\mathbf{HC}(\phi(s))}) = 1]\Big|)\\
&= \frac{1}{2} + \frac{1}{2}\cdot\Big|\Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \mathbf{HC}(\phi(s))) = 1]\\
&\quad - \Pr_{s\in\{0,1\}^\lambda}[\mathcal{D}(\mathbf{Eval_F}(k,\phi(s)), \overline{\mathbf{HC}(\phi(s))}) = 1]\Big|\\
&= \frac{1}{2} + \varepsilon(\lambda).
\end{aligned}$$

So $\mathcal{A}$ guesses $\mathbf{HC}(\phi(s))$ with probability $\frac{1}{2} + \varepsilon(\lambda)$. We get a contradiction to the assumption that $\mathbf{HC}$ is a hardcore function for arbitrarily correlated inputs since $\varepsilon(\lambda)$ is a non-negligible. We complete the proof.

B. RRA-CPA SECURE PKE SCHEMES VIA wRKAOWF

Our construction is the same as the PKE construction from indistinguishability obfuscation in [31]. We prove that if the PRG used in the construction is $\Phi-$related-seed secure, then the PKE scheme is RRA-CPA secure against a Φ-restricted adversary. Let PRG be a $\Phi-$related-seed secure pseudorandom generator that which takes a seed $r \in \{0,1\}^\lambda$ and outputs a pseudorandom value $prv \in \{0,1\}^{2\lambda}$. Let F be a puncturable PRF whose domain is $\{0,1\}^{2\lambda}$ and range $\subseteq \{0,1\}^l$. The construction is described as follows:

- Setup(1^λ): Chooses a puncturable PRF key K for F as the secret key SK and obfuscates the program PKE Encrypt in Table IV. The public key, PK, is the obfuscated program. PKE Encrypt* in Table 4 is only used in the proof of security.
- Encrypt($PK, m \in \mathcal{M}$): Runs the public key PK which is an obfuscated program taking as input a random coin $r \in \{0,1\}^\lambda$ and the plaintext m.
- Decrypt($SK, c = (c_1, c_2)$): Outputs $m' = F(K, c_1) \bigoplus c_2$.

TABLE 4. PKE encrypt and PKE encrypt*.

PKE Encrypt:
Hardwired:Key of the punctured PRF, i.e., K.
Input parameters: Plaintext $m \in \{0,1\}^l$, random value $r \in \{0,1\}^\lambda$.
1. Compute $prv = PRG(r)$
2. Output $c = (c_1 = prv, c_2 = F(K, prv) \bigoplus m)$

PKE Encrypt*:
Hardwired:Key of the punctured PRF, i.e., $K(\{prv^*\})$.
Input parameters: Plaintext $m \in \{0,1\}^l$, random value $r \in \{0,1\}^\lambda$.
1. Compute $prv = PRG(r)$
2. Output $c = (c_1 = prv, c_2 = F(K, prv) \bigoplus m)$

Proof: In [31], the construction is proved IND-CPA secure by a sequence of hybrid games Hyb_0,Hyb_1,Hyb_2,Hyb_3. As we only change the pseudorandomness generator PRG in the construction of [31] to a $\Phi-$related-seed secure pseudorandomness generator, Φ-RRA-CPA security of the construction can be proved using the same technique while just modifying the first game Hyb_0 as follow:

- Hyb_0: This is the original RRA-CPA security game.
  1) $r^* \in \{0,1\}^\lambda$ is chosen randomly and $prv^* = PRG(\phi(r^*))$.
  2) The key for the puncturable PRF is set to K.
  3) The obfuscated program PKE Encrypt is taken as the public key PK.
  4) The adversary receives PK and sends to the challenger $m_0, m_1 \in \{0,1\}^l$.
  5) The challenger outputs the challenge ciphertext $c^* = (c_1^* = prv^*, c_2^* = F(K, prv^*) \bigoplus m_b)$ by randomly choosing $b \in \{0,1\}$.

The description of Hyb_1,Hyb_2,Hyb_3 is straightly same as that in [31].

- Hyb_1: this game only changes the generation step of prv^* in Hyb_0 by choosing it randomly in $\{0,1\}^{2\lambda}$.
- Hyb_2: this game only changes the public key generation step in Hyb_1. The public key in Hyb_2 is generated by obfuscating the program PKE Encrypt* in Table IV.
- Hyb_3: compared with Hyb_2, this game only changes the challenge ciphertext. A random z^* is chosen and the challenge ciphertext is formed as $(c_1^* = prv^*, c_2^* = z^*)$.

As PRG is $\Phi-$related-seed secure, we can prove Hyb_0 and Hyb_1 is indistinguishable. Also, the indistinguishabil-

TABLE 5. Performance.

g	p	x	speed
6703903964971298	7079874630119430	6982670607535038	0.698
5497870124991029	7966473239241332	3608333877035355	ms
2306373968291029	3184040858869399	2250794702498809	(512
6196688861780721	3280009524913780	1051400140924913	bit)
8608820150367734	1843937873039955	2068568074704133	
8840093714908345	5094951199638072	8205516019159196	
1713845015929093	6140939419197764	8239180861084026	
2430254268769414	7860661263074023	5053520803427503	
0597328497321682	3484704698603189	5478352008077829	
4503042048	6530667173	1923285934	
8988465674311579	1347323977591077	1215943646403727	3.789
5386465259539451	8468954726086764	4756578045615499	ms
2366808988489471	6234257252180822	7162661273709810	(1024
1532863671504057	9146745758754917	4503625284538636	bit)
8866337902750481	1487070997735519	8096054132811714	
5663542386612037	0117010489662127	8136127537244600	
6801056005693993	4186485799003385	3274003843283683	
5696678829394884	6745222265605295	2860911388172167	
4072083112464237	2584830279506967	8691383336164017	
1531973706218888	8036661124596968	6119872325088536	
3946712432742638	3976657243877795	7771685167886980	
1511098006230470	3486722322262059	7341259065510667	
5972654147604250	8137187907838680	8416217085676861	
2884419075341171	9339259688846580	7709295432825500	
2314407369565552	4646398431124100	2428765548895876	
7041361858167525	2839677757678194	0918997898865297	
5342293149119973	6875449725602808	7916883683807098	
6229692398581524	5548213521126993	8584881670620069	
1767816481211206	4186050468771269	9528819627828169	
8608	64667	15523	
161585030356555	299361057048769	266899595314769	24.14
036503574383443	065525756228086	720105788588627	ms
349759802220513	759647571412168	716292430602128	(2048
348577420160651	837771367599783	571569920303215	bit)
727137623275694	118511086709852	134074370227890	
339454465986007	369635333481260	511589520949543	
057614567318443	163074838875963	968416743600542	
589804609490097	651390750566000	700838012027553	
470597795752454	439452557560531	856566900756490	
605475440761932	237032529615219	723684589326528	
241415603154386	944423378938453	688114668841736	
836504980458750	117430703022562	193390909006273	
988751948260533	973019204267249	159318846526370	
980288191920337	467698212422981	389476254300821	
841383961093213	386168323786868	918135529820936	
098780809190471	937052028929146	833486393413414	
692380852352908	665017826858067	081224267517238	
229260181525214	311749448025650	175239827964655	
437879457705329	759452857714863	855387811639446	
043037761995619	339079752942722	706109445107618	
651927609571666	679625792098082	197452855521467	
948341712103424	530965000282889	156386273073696	
873932822847474	881322834100124	457143143234781	
280880176631610	779799233544839	050513565201690	
290389028296655	234768890063892	226253840254670	
130963542301570	384478930427402	519421646763123	
751292964320885	546525238877878	189373559644169	
583629718018592	533300493898628	616922486280164	
309286787991755	415778152369610	821612807674365	
761508229522018	685317348233871	284623327552980	
488066166436156	705843953632808	152374921943878	
135628423554101	508347865854863	919858794540196	
048625785508634	384263034324843	225473454321771	
656617348392712	161940038796420	224183180167594	
903283489675229	202891662384897	930698983363684	
986341764993191	557567600816077	878972795195219	
077625831947186	317531987912467	992752188389466	
677718010677166	927940284612649	913014885965814	
148023226592393	878278384620814	762501850418095	
024760740967779	288431527322652	753396588600989	
268055297981153	892833717219856	876467357041738	
28	53	38	

ity between Hyb_1 and Hyb_2, and the indistinguishability between Hyb_2 and Hyb_3 were proved in [11]. As the adversary has zero advantage in Hyb_3, we prove the Φ-RRA-CPA security of the construction.

C. PERFORMANCE

We will give some concrete performance results for the above scheme. As the encryption algorithm has to execute an obfuscated program, we must admit its inefficiency in real life. Therefore, we only show the efficiency of the decryption algorithm of our scheme. We use Python to generate three groups with their generator g and order p in the length of 512/1024/2048 bits and formalize one-way functions $\mathbf{F}$ respectively. To evaluate the somewhat worst-case performance, we generate keys for these $\mathbf{F}$s in the same bit length as its respective order p. The following Table 5 gives the results, where the decryption speed is tested on a personal computer with Intel(R) Core(TM) i7-8550 CPU @ 1.80GHz, 8GB RAM and Windows 10 operating system.

IV. RRA-SECURE PKE SCHEME AGAINST ARBITRARY FUNCTION

A. CONSTRUCTION FROM PUBLICLY DENIABLE ENCRYPTION

The first construction comes from the publicly deniable encryption. We will prove that any publicly deniable encryption scheme is RRA-secure PKE scheme against arbitrary function. Publicly deniable encryption is formally proposed by [31] as follows:

Definition 6: (Publicly Deniable Encryption). Let PDE Π = (Setup, Encrypt, Decrypt, Explain) be a publicly deniable encryption scheme.

- Setup(1^λ): it is a randomized algorithm which outputs a public/secret key pair (pk,sk) taking as input a security parameter λ.
- Encrypt($pk, m; u$): it is a probabilistic algorithm which uses a public key pk and random coins u to encrypt a plaintext m to its corresponding ciphertext c.
- Decrypt(sk, c): it is a deterministic algorithm which can obtain the corresponding plaintext m or error symbol $\perp$ by decrypting a ciphertext c using a secret key sk.
- Explain($pk, c, m; r$): it can output a randomness e which is with the same size as the random coin u used in Encrypt above given a public key pk, a ciphertext c, and a plaintext m.

In [31], two security requirements for publicly deniable encryption called *Indistinguishability under Chosen Plaintext Attack*(IND-CPA) and *Indistinguishability of Explanation* are presented as follows:

Indistinguishability under Chosen Plaintext Attack. For any probabilistic polynomial-time adversary of a publicly deniable encryption PDE Π, $\mathcal{A}$, the following advantage $\text{Adv}^{IND-CPA}_{\Pi,\mathcal{A}}(\lambda)$ is a negligible function in n.

$$\text{Adv}^{IND-CPA}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l| l} & (sk, pk) \leftarrow Setup(1^\lambda); \\ & (m_0, m_1) \leftarrow \mathcal{A}; \\ b = b' \ : & b \leftarrow_R \{0,1\}; \\ & c^* \leftarrow Encrypt(pk, m_b; u); \\ & b' \leftarrow \mathcal{A}(c^*) \end{array}\right] - \frac{1}{2}$$

TABLE 6. Enc and LR oracles in game_1.

$\underline{\text{EO}(pk, m, \phi)}$:	$\underline{\text{LR}(m_0, m_1, i, \phi)}$:
If CoinTable[i]=$\perp$	If CoinTable[i]=$\perp$
CoinTable[i]$\leftarrow_R$**Rnd**	CoinTable[i]$\leftarrow_R$**Rnd**
$r_i \leftarrow$CoinTable[i]	$r_i \leftarrow$CoinTable[i]
$c \leftarrow$PKE.Encrypt$(pk, m, \phi(r_i))$	$c^* \leftarrow$PKE.Encrypt$(pk^*, m_b, \phi(r_i))$
$r' =$PKE.Explain$(pk, c, m; u)$	$r^{'} =$PKE.Explain$(pk^*, c, m; u)$
$c' =$PKE.Encrypt$(pk, m, r^{'})$	$c'^* =$PKE.Encrypt$(pk^*, m, r^{'})$
Return c'	Return c'^*

Indistinguishability of Explanation. For any probabilistic polynomial-time adversary of a publicly deniable encryption PDE Π, $\mathcal{A}$, the following advantage $\text{Adv}^{IND-EXP}_{\Pi,\mathcal{A}}(\lambda)$ is a negligible function in n.

$$\text{Adv}^{IND-EXP}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[b = b' : \begin{array}{l} (sk, pk) \leftarrow Setup(1^\lambda); \\ m \leftarrow \mathcal{A}; \\ c^* \leftarrow Encrypt(pk, m; u_0); \\ u_1 = Explain(pk, c, m; r); \\ b \leftarrow_R \{0, 1\}; \\ b' \leftarrow \mathcal{A}(c^*, u_b) \end{array} \right] - \frac{1}{2}$$

Theorem 3: If the publicly deniable encryption scheme satisfies *Indistinguishability under Chosen Plainext Attack* and *Indistinguishability of Explanation*, then this scheme is RRA-CPA secure against arbitrary function.

Proof: We can prove the above theorem via a sequence of games.

Game_0: This is the real RRA-CPA security game played by an adversary $\mathcal{A}$ against the publicly deniable encryption scheme.

Game_1: This game is the same as Game_0 except that we explain the ciphertext c and c^* to get a new randomness and return a new ciphertext obtained by encrypting the message using this new randomness. More precisely, in this game, the Enc and LR oracle are changed as follows in Table VI:

The *Indistinguishability of Explanation* of the publicly deniable encryption scheme guarantees that Game_0 and Game_1 are computationally indistinguishable.

Game_2: This game is the same as Game_1 except that r' in Enc and LR oracle is randomly selected from **Rnd**. Also, the *Indistinguishability of Explanation* of the publicly deniable encryption scheme guarantees that Game_1 and Game_2 are computationally indistinguishable.

In Game_2, the *Indistinguishability under Chosen Plainext Attack* of the publicly deniable encryption scheme guarantees that the adversary's advantage to break the security of the scheme is negligible. Therefore, we can prove the theorem as the advantage of the adversary in real RRA-CPA security game is negligible by the indistinguishability of all the games.

TABLE 7. Scheme HC-PKE based on a standard PKE scheme, PKE and a hardcore function for correlated inputs, HC.

$\underline{\text{Alg. HC-PKE.KeyGen}(1^\lambda)}$:	$\underline{\text{Alg. HC-PKE.Encrypt}(pk, m)}$:
$(pk, sk) \leftarrow$PKE.KeyGen(1^λ)	$r \leftarrow$ **Rnd**
$\underline{\text{Alg. HC-PKE.Decrypt}(sk, c)}$:	$r' \leftarrow$HC(r)
$m \leftarrow$PKE.Decrypt(sk,c)	$c \leftarrow$PKE.Encrypt$(pk, m; r')$
return m	return c

TABLE 8. Enc and LR oracles in game_0.

$\underline{\text{EO}(pk, m, \phi)}$:	$\underline{\text{LR}(m_0, m_1, i, \phi)}$:
If CoinTable[i]=$\perp$	If CoinTable[i]=$\perp$
CoinTable[i]$\leftarrow_R$**Rnd**	CoinTable[i]$\leftarrow_R$**Rnd**
$r_i \leftarrow$CoinTable[i]	$r_i \leftarrow$CoinTable[i]
$r' = \mathbf{HC}(\phi(r_i))$	$r' = \mathbf{HC}(\phi(r_i))$
$c' \leftarrow$PKE.Encrypt$(pk, m; r')$	$c'^* \leftarrow$PKE.Encrypt$(pk^*, m_b; r')$
Return c'	Return c'^*

TABLE 9. EO and LR oracles in game_1.

$\underline{\text{Enc}(pk, m, \phi)}$:	$\underline{\text{LR}(m_0, m_1, i, \phi)}$:
If CoinTable[i]=$\perp$	If CoinTable[i]=$\perp$
CoinTable[i]$\leftarrow_R$**Rnd**	CoinTable[i]$\leftarrow_R$**Rnd**
$r_i \leftarrow$CoinTable[i]	$r_i \leftarrow$CoinTable[i]
$r' \leftarrow_R$**Rnd**	$r^{'} \leftarrow_R$**Rnd**
$c' =$PKE.Encrypt$(pk, m; r')$	$c'^* =$PKE.Encrypt$(pk^*, m; r')$
Return c'	Return c'^*

B. CONSTRUCTION FROM POLY-MANY HARDCORE BITS FOR ARBITRARILY CORRELATED INPUTS

The above construction is only secure under Chosen-Plainext-Attack. In the following, we will describe the construction of a RRA-CCA secure public-key encryption scheme. The construction combines a hardcore function for arbitrarily correlated inputs [30] with an IND-CCA secure PKE scheme. Specifically, the randomness r is used as an input to the hardcore function, the output from the hardcore function is then used as the actual randomness for encryption. Table 7 formalizes the construction, and we prove its security in the following theorem.

Theorem 4: If **HC** is hardcore function for arbitrarily correlated inputs, then the scheme in Table 7 is RRA-ATK secure against a Φ-restricted adversary where Φ is a class of arbitrary functions.

Proof: We can prove the above theorem via two games.

Game_0: This is the real RRA-ATK security game played by an adversary $\mathcal{A}$ against the scheme in Table VI. The Enc and LR oracles are shown in Table VIII.

Game_1: This game is the same as Game_0 except that r' in Enc and LR oracle is randomly selected from **Rnd** as in Table IX:

The hardcore function **HC** for arbitrarily correlated inputs guarantees that Game_0 and Game_1 are computationally indistinguishable.

In Game_1, the *IND-ATK* security of the public-key encryption scheme guarantees that the adversary's advantage to break the security of the scheme is negligible.

Therefore, we can prove the theorem as the advantage of the adversary in real RRA-ATK security game is negligible by the indistinguishability of all the games.

C. PERFORMANCE ANALYSIS

The two constructions are based on PDE and PKE schemes respectively. The first construction is as efficient as the based PDE scheme. The efficiency of the second construction relies on the based PKE scheme and the hardcore function, and the execution time is the total time of both of them.

V. CONCLUSION

In this paper, we give some methods of constructing a secure PKE scheme against related randomness attacks. We propose a RRA-CPA secure PKE scheme with an efficient decryption algorithm and short ciphertexts size. To obtain RRA-secure PKE scheme against arbitrary function, we first prove that any publicly deniable encryption scheme is a RRA-CPA secure public-key encryption scheme against arbitrary function. Then we combine standard IND-CCA PKE scheme with hardcore function for arbitrarily correlated inputs to get a RRA-CCA secure public-key encryption scheme against arbitrary function. In terms of efficiency, the encryption algorithm of our first proposed scheme is inefficient, while the decryption algorithm of it is very efficient. Our first scheme secure against arbitrary function is actually a publicly deniable encryption scheme so that it is inefficient at present, as the known publicly deniable encryption schemes are constructed based on indistinguishability obfuscation which is not practical at this stage. We have to admit that use of indistinguishability obfuscation in the first two schemes of our work makes the methods only with theoretical significance. Their practical significance depends on the development of indistinguishability obfuscation in efficiency. Compared with it, the efficiency of our second scheme secure against any arbitrary function depends on the PKE scheme and hardcore function it is based on which is acceptable. In the future, we will measure the performance of our proposed methods via the cloud-edge platform and study how to construct other cryptographic primitives with RRA-Security such as IBE, ABE, and so on.

REFERENCES

[1] L. Qi, X. Zhang, W. Dou, and Q. Ni, "A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2616–2624, Nov. 2017.

[2] Y. Xu, L. Qi, W. Dou, and J. Yu, "Privacy-preserving and scalable service recommendation based on SimHash in a distributed cloud environment," *Complexity*, vol. 2017, pp. 1–9, 2017.

[3] C. Yan, X. Cui, L. Qi, X. Xu, and X. Zhang, "Privacy-aware data publishing and integration for collaborative service recommendation," *IEEE Access*, vol. 6, pp. 43021–43028, 2018.

[4] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, "Ensuring security and privacy preservation for cloud data services," *ACM Comput. Surv.*, vol. 49, no. 1, pp. 1–39, Jun. 2016.

[5] Y. Wang, M. Zhao, Y. Hu, Y. Gao, and X. Cui, "Secure computation protocols under asymmetric scenarios in enterprise information system," *Enterprise Inf. Syst.*, pp. 1–21, Mar. 2019, doi: 10.1080/17517575.2019.1597387.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[7] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 80–85, Nov. 2017.

[8] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, Nov. 2017.

[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[10] L. T. Yang, X. Wang, X. Chen, L. Wang, R. Ranjan, X. Chen, and M. J. Deen, "A multi-order distributed HOSVD with its incremental computing for big services in cyber-physical-social systems," *IEEE Trans. Big Data*, to be published, doi: 10.1109/tbdata.2018.2824303.

[11] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, and J. Chen, "A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment," *Future Gener. Comput. Syst.*, vol. 88, pp. 636–643, Nov. 2018.

[12] W. Gong, L. Qi, and Y. Xu, "Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–8, 2018.

[13] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, "Time-aware distributed service recommendation with privacy-preservation," *Inf. Sci.*, vol. 480, pp. 354–364, Apr. 2019.

[14] X. Wang, L. T. Yang, L. Kuang, X. Liu, Q. Zhang, and M. J. Deen, "A tensor-based big-data-driven routing recommendation approach for heterogeneous networks," *IEEE Netw.*, vol. 33, no. 1, pp. 64–69, Jan. 2019.

[15] L. Qi, X. Zhang, S. Li, S. Wan, Y. Wen, and W. Gong, "Spatial-temporal data-driven service recommendation with privacy-preservation," *Inf. Sci.*, vol. 515, pp. 91–102, Apr. 2020.

[16] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.

[17] X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, and B. O. Apduhan, "NQA: A nested anti-collision algorithm for RFID systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 4, pp. 1–21, Jul. 2019.

[18] Y. Dodis, D. Pointcheval, S. Ruhault, D. Vergniaud, and D. Wichs, "Security analysis of pseudo-random number generators with input: /Dev/random is not robust," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Berlin, Germany, 2013, pp. 647–658.

[19] L. Dorrendorf, Z. Gutterman, and B. Pinkas, "Cryptanalysis of the windows random number generator," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, Alexandria, VA, USA, 2007, pp. 476–485.

[20] K. Michaelis, C. Meyer, and J. Schwenk, "Randomly failed! The state of randomness in current java implementations," in *Proc. CT-RSA*, San Francisco, CA, USA, 2013, pp. 129–144.

[21] Y. Wang, A. Bracciali, T. Li, F. Li, X. Cui, and M. Zhao, "Randomness invalidates criminal smart contracts," *Inf. Sci.*, vol. 477, pp. 291–301, Mar. 2019.

[22] T. Ristenpart and S. Yilek, "When good randomness Goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptograph," in *Proc. NDSS*, San Diego, CA, USA, 2010, pp. 1–18.

[23] P. Austrin, K.-M. Chung, M. Mahmoody, R. Pass, and K. Seth, "On the impossibility of cryptography with tamperable randomness," *Algorithmica*, vol. 79, no. 4, pp. 1052–1101, Dec. 2017.

[24] M. Feltz and C. Cremers, "Strengthening the security of authenticated key exchange against bad randomness," *Des., Codes Cryptogr.*, vol. 86, no. 3, pp. 481–516, Mar. 2018.

[25] K. G. Paterson, J. C. N. Schuldt, and D. L. Sibborn, "Related randomness attacks for public key encryption," in *Proc. PKC*, Buenos Aires, Argentina, 2014, pp. 465–482.

[26] V. Goyal, A. O'Neill, and V. Rao, "Correlated-input secure hash functions," in *Proc. TCC*, Providence, RI, USA, 2011, pp. 182–200.

[27] K. G. Paterson, J. C. N. Schuldt, D. L. Sibborn, and H. Wee, "Security against related randomness attacks via reconstructive extractors," in *Proc. IMACC*, Oxford, U.K., 2015, pp. 23–40.

[28] T. H. Yuen, C. Zhang, S. S. Chow, and S. M. Yiu, "Related randomness attacks for public key cryptosystems," in *Proc. 10th ACM Symp. Inf., Comput. Commun. Secur. (ASIA CCS)*, 2015, pp. 215–223.

[29] J. C. Schuldt and K. Shinagawa, "On the robustness of RSA-OAEP encryption and RSA-PSS signatures against (malicious) randomness failures," in *Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS)*, Abu Dhabi, UAE, 2017, pp. 241–252.
[30] M. Bellare, I. Stepanovs, and S. Tessaro, "Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation," in *Proc. ASIACRYPT*, Kaohsiung, Taiwan, 2014, pp. 102–121.
[31] A. Sahai and B. Waters, "How to use indistinguishability obfuscation: Deniable encryption, and more," in *Proc. 46th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 2014, pp. 475–484.
[32] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proc. 32nd Annu. ACM Symp. Theory Comput. (STOC)*, Baltimore, MD, USA, 1990, pp. 427–437.
[33] M. Bellare, S. Meiklejohn, and S. Thomson, "Key-versatile signatures and applications: RKA, KDM and joint enc/sig," in *Proc. EUROCRYPT*, Copenhagen, Denmark, 2014, pp. 496–513.
[34] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proc. FOCS*, Berkeley, CA, USA, 2013, pp. 40–49.
[35] M. Ben-Or, "Probabilistic algorithms in finite fields," in *Proc. FOCS*, Nashville, TN, USA, 1981, pp. 394–398.

PENGTAO LIU received the master's degree in computer software and theory from Shandong University. She is currently with the College of Cybersecurity, Shandong University of Political Science and Law. Her main research interests include public key encryption and digital signature.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 21, 2019, accepted March 11, 2019, date of publication March 15, 2019, date of current version April 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905250

Public-Key Encryption With Keyword Search via Obfuscation

CHENGYU HU[1], PENGTAO LIU[2], RUPENG YANG[3], AND YAN XU[4]
[1]School of Software, Shandong University, Jinan 250101, China
[2]School of Cyberspace Security, Shandong University of Political Science and Law, Jinan 250014, China
[3]School of Computer Science and Technology, Shandong University, Jinan 250101, China
[4]School of Computer Science and Technology, Anhui University, Hefei 230601, China

Corresponding author: Chengyu Hu (hcy@sdu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602275 and Grant 61772311, in part by the Shandong Province Higher Educational Science and Technology Program under Grant J15LN01, and in part by the Open Project of Co-Innovation Center for Information Supply and Assurance Technology, Anhui University, under Grant ADXXBZ201702.

ABSTRACT *Public-key encryption with keyword search (PEKS)* enables users to search on encrypted data, which is applicable to the scenario of sharing data in the cloud storage. In this paper, we focus on how to construct a PEKS scheme via obfuscation. Our basic scheme is built on the differing-inputs obfuscation (diO) and can be considered as an initial attempt to apply diO in the PEKS field. The scheme supports searching on encrypted data by providing to the cloud server an obfuscated simple ''decrypt-then-compare'' circuit with the secret key and the queried keyword hardwired in it. More interestingly, the scheme can be simply improved to resist off-line keyword guessing attacks (KGAs) as the standard PEKS scheme rather than a designated tester one. For complex search conditions, our scheme can be easily extended to multiple functionalities, such as conjunctive and fuzzy keyword search. Furthermore, it can be extended to the PEKS scheme in the multi-user setting.

INDEX TERMS Public-key encryption with keyword search, obfuscation, keyword guessing attacks, multiple functionalities, multi-user setting.

I. INTRODUCTION

As cloud can provide elastic and highly available storage service, many users decide to outsource their data in the cloud storage with the basic requirement that the cloud should supply some mechanisms, such as access control [1], [2], trust management [3], [4], auditing [5]–[8], etc. Further, as the data owners lose full control of the outsourced data and cannot prevent malicious cloud server from reading their data, they have to encrypt the sensitive data before outsourcing it to the cloud. By this way, confidentiality of the data is guaranteed. However, as the cloud server cannot read the original data, it causes a problem of searching over the encrypted data without decrypting it. To resolve this problem, some encrypted database management systems [9], [10] have been proposed while some other works focused on the keyword-based search on encrypted data which is also called ''searchable encryption''. The notion of searchable encryption is proposed and studied in two lines. One is in private key model like [11]–[20], the other is in public key model, i.e., the concept of Public-Key Encryption with Keyword Search (PEKS). In a PEKS scheme, a sender uploads encrypted data to a server along with a list of ciphertexts for each keyword associated with the data encrypted under a receiver's public key. Then, the receiver can send to the server a trapdoor of a keyword which can be used by the server to test the encrypted keywords list, and obtain the matched index of the data containing the keyword from the server.

The associate editor coordinating the review of this manuscript and approving it for publication was Ramakrishnan Srinivasan.

A. RELATED WORKS AND OUR CONTRIBUTIONS

Since the initial PEKS scheme [21] constructed from identity-based encryption (IBE) [22], there have been a lot of researches emerging to enhance the PEKS. Abdalla *et al.* [23] showed that it is the anonymity of IBE that ensures a PEKS ciphertext reveals no information about the keyword and presented an improved universal transformation from anonymous IBE to PEKS. In functionality, there are various PEKS schemes proposed to search on complex conditions. For examples, Boneh and Waters [24] proposed a

PEKS scheme which supports conjunctive, subset, and range comparisons over the keywords. Sedghi *et al.* [25] presented a PEKS scheme which supports queries for keywords containing wildcards based on hidden vector encryption scheme. Lai *et al.* presented a more expressive and efficient construction in [26]. A new notion of attribute-based keyword search in encrypted data was studied in [27] and [28], and the works in [29]–[34] studied the severe vulnerability of PEKS to the off-line keyword guessing attacks (KGA).

The KGA attack was first introduced by Byun *et al.* [29] depending on the fact that the keyword space is somewhat small in the real world. In an off-line keyword guessing attack, the attacker (a malicious user or malicious server) can generate the PEKS ciphertexts for some keywords and test the obtained trapdoor with the ciphertexts to get the information about the keyword of the trapdoor. Byun *et al.* [29] left an open problem to construct secure PEKS schemes against keyword guessing attacks. A well-known method to solve this problem is introducing a modified definition of PEKS called ''searchable public key encryption with a designated tester (dPEKS)'' [31] in which the server's public key is used in the keyword ciphertext and trapdoor generation algorithms so that only the server can execute the test algorithm correctly using its corresponding secret key. In this way, dPEKS schemes can only resist keyword guessing attacks performed by malicious users. To prevent a malicious server from guessing the keyword, Huang and Li [34] proposed ''Public-key Authenticated Encryption with Keyword Search (PAEKS)''. However, in their scheme, the trapdoor for a keyword keeps the same, which does not satisfy ''*trapdoor indistinguishability*''. Meanwhile, it does not satisfy ''*ciphertext indistinguishability*''. Actually, a malicious server can figure out the keyword associated with a keyword ciphertext without a trapdoor generated by the receiver.

As data will always be shared between users of the cloud, it is desiderated to construct PEKS schemes in the multi-user setting. The trivial method to construct a PEKS scheme in the multi-user setting is that the data owner encrypts keywords for each receiver which increases burdens of the data owner and the communication overheads. To solve the problem, there are mainly two kinds of measures according to whether or not a trusted third party (TTP) exists. The TTP manages the user's key which helps a user to generate the trapdoor when the user wants to search on the encrypted data from a data owner [35]–[37]. The schemes without TTP usually either need to predetermine the users and pre-compute some parameters by all the users [38]–[40], or let the data owner play a role of TTP [41].

1) OUR CONTRIBUTIONS

In this paper, we focus on how to construct a public-key encryption with keyword search scheme (PEKS) via obfuscation. To our delight, we find that *differing-inputs obfuscation* [42] can be used to build PEKS schemes. We apply differing-inputs obfuscation in PEKS field and obtain several interesting theoretical results.[1] We summarize our contributions as follows:

- Adopt differing-inputs obfuscation to design a new PEKS scheme with single-keyword search. We set the keyword ciphertext as the ciphertext of the keyword encrypted by a standard public-key encryption (PKE) scheme under the receiver's public key and set the trapdoor as the obfuscated ''decrypt-then-compare'' circuit with the secret key and the queried keyword hardwired in it. To make it succeed in proving the security, we use the ''double encryption'' paradigm [43] in constructing the program to be obfuscated.
- Improve the efficiency of our basic PEKS scheme and extend it to resist off-line keyword guessing attacks. We slightly change the implementation model of PEKS, i.e., the receiver generates an obfuscated circuit of the search circuit and sends it to the tester only once. Meanwhile, a trapdoor for a keyword w is set as double encryptions of it under the receiver's public key. The trapdoor also contains a signature of the keyword w under the receiver's signing key to guarantee that only the receiver can generate a valid trapdoor. To make our scheme secure against keyword guessing attacks, a signature of the keyword under the sender's signing key is added in the keyword ciphertext to ensure that the tester cannot generate a valid keyword ciphertext of a guessing keyword and test it with a trapdoor. Compared with dPEKS schemes resisting KGA attacks, our scheme is a standard PEKS scheme rather than a PEKS scheme with a designated tester, i.e., our scheme can resist the malicious server, and any tester who obtains the keyword ciphertext and the trapdoor can execute the test algorithm.
- Extend our PEKS scheme to that in the multi-user setting. In the scheme, the sender can provide the tester with an obfuscated circuit which makes a re-encryption of a keyword ciphertext destined for user i and turns it into that for user j. Then the tester can implement keyword search on the keyword ciphertext for user j with the trapdoors generated by user j. Compared with other PEKS schemes in the multi-user setting, our scheme neither needs to predetermine all users and pre-compute keyword ciphertexts for all users [38], [39], nor needs a cooperative computation process [40] or a trusted third party to manage the user's key [35]–[37].
- Our scheme can be easily extended to schemes which support other functionalities, such as multiple keywords search, fuzzy keyword search, etc.

[1] As far as we know, until now, there is no secure cryptographic obfuscation which can be efficiently implemented in real life. Nevertheless, we show that it can be used to construct PEKS schemes with good properties which can not be easily achieved. Once cryptographic obfuscation can be efficiently implemented in the future, our resulting PEKS schemes will be used in practice.

2) REMARK

As our proposed PEKS schemes are based on cryptographic obfuscation, the usability of them depends on the efficiency and security of the cryptographic obfuscation schemes. We have to admit that our schemes are not implementable currently as a whole and we cannot supply full experimental performance of our scheme. This is mainly because we still do not know how to construct a secure cryptographic obfuscation until now. There are a few candidates, but to the best of our knowledge, they are all broken. Actually, there are several works about other cryptographic primitives based on obfuscation, such as functional encryption [44], deniable encryption [45], verifiable searchable symmetric encryption [46], proofs of retrievability [47], multi-party key exchange [48], [49], proxy re-encryption [50], [51]. All these works have no experimental result about the concrete performance due to the same reason. Nevertheless, our schemes can be used in practice once cryptographic obfuscation can be efficiently implemented in the future. Thus, we believe that our work is of theoretical significance and is of potentially practical significance.

B. ORGANIZATION

In Section 2, we review some preliminaries. In Section 3, we give the basic construction of our PEKS scheme and extend it to that secure against off-line keyword guessing attacks. We give a theoretical analysis of our extended scheme and show how to construct a PEKS scheme supporting complex functionalities based on it. We also show how to extend it to that in the multi-user setting. Finally, we draw our conclusions in Section 4.

II. PRELIMINARIES

In this section, we recall some basic notions, terminologies.

A. PUBLIC KEY ENCRYPTION

Public key encryption (PKE) scheme [43] consists of three polynomial-time algorithms denoted by PKE = (PKE.KeyGen,PKE.Enc, PKE.Dec). The randomized key generation algorithm PKE.KeyGen takes the security parameter as its input and outputs a public/secret key pair (pk, sk). The probabilistic encryption algorithm PKE.Enc, on inputs a message $m \in \mathcal{M}$ and a public key pk, chooses random coins from **Rnd** and uses these coins to output a ciphertext c. The deterministic decryption algorithm, on inputs a ciphertext c and a secret key sk, either outputs a message m or an error symbol $\perp$.

1) IND-ATK SECURITY (ATK=CPA,CCA)

The *Indistinguishability under Chosen Plaintext Attack, IND-CPA* and *Indistinguishability under Chosen Ciphertext Attack, IND-CCA* are two security definitions of public key encryption. More formally, for any probabilistic polynomial-time IND-ATK adversary of a PKE Π, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the following advantage $\mathrm{Adv}^{IND-ATK}_{\Pi,\mathcal{A}}(\lambda)$ of adversary is a negligible function in λ. (If ATK = CPA, the adversary's access to Dec oracle is removed.)

$$Adv^{IND-ATK}_{\Pi,\mathcal{A}}(\lambda) = Pr\left[\begin{array}{ll} b = b' : & (sk, pk) \leftarrow KeyGen(1^\lambda); \\ & (state, m_0, m_1) \\ & \quad \leftarrow \mathcal{A}_1^{Enc(pk,m),Dec(sk,c)}(\cdot); \\ & b \leftarrow_R \{0,1\}; \\ & c^* \leftarrow LR(m_0, m_1); \\ & b' \leftarrow \mathcal{A}_2^{Enc(pk,m),Dec(sk,c)}(pk, c^*, state) \end{array}\right] - \frac{1}{2}$$

B. SIGNATURE

A signature scheme [52] is a tuple of probabilistic polynomial-time algorithms (KeyGen,Sig,Ver) satisfying the following:

1) The key-generation algorithm KeyGen takes as input a security parameter λ and outputs the public/signing key pair (pk, sk).
2) The signing algorithm Sig takes as input a signing key sk and a message m from some underlying message space. It outputs a signature $\sigma \leftarrow \mathrm{Sign}(sk, m)$.
3) The *deterministic* verification algorithm Ver takes as input a public key pk, a message m, and a signature σ. It outputs 1 meaning valid and 0 meaning invalid.

1) EUF-CMA SECURITY

A signature scheme SIG=(KeyGen, Sig,Ver) is *existentially unforgeable under an adaptive chosen-message attack* (EUF-CMA) if for any probabilistic polynomial-time adversary $\mathcal{A}$, the following advantage $\mathrm{Adv}^{EUF-CMA}_{\Pi,\mathcal{A}}(\lambda)$ of the adversary is a negligible function in λ.

$$Adv^{EUF-CMA}_{SIG,\mathcal{A}}(\lambda) = Pr\left[\begin{array}{ll} Ver(m^*, \sigma^*) = 1 \\ m^* \notin \mathcal{Q} \end{array} : \begin{array}{l} \mathcal{Q} = \emptyset, (sk, pk) \leftarrow KeyGen(\lambda); \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{Sig(sk,m,\mathcal{Q})}(\cdot); \end{array}\right]$$

The signing oracle $\mathrm{Sig}(sk, m, \mathcal{Q})$ works as following:

$\mathbf{Sig}(sk, m, \mathcal{Q})$:
1. σ=Sig(sk, m);
2. set $\mathcal{Q} = \mathcal{Q} \bigcup m$;
3. return σ.

C. NON-INTERACTIVE ZERO-KNOWLEDGE

Let R be an NP relation on pairs (x, y) with corresponding language $L_R = \{y | \exists x s.t. (x, y) \in R\}$. A non-interactive zero-knowledge (NIZK) argument for a relation R [53] consists of three algorithms (Setup,$\mathcal{P}$,$\mathcal{V}$) with syntax:

1) The parameter setup algorithm Setup takes as input a security parameter λ and outputs a common reference string (CRS).
2) The prove algorithm $\mathcal{P}$ takes as input y, the witness x of $y \in L_R$, and creates an argument $\pi \leftarrow \mathcal{P}_{CRS}(x, y)$ that $R(x, y) = 1$.

3) The verification algorithm $\mathcal{V}$ takes as input y and an argument π, and outputs 1/0 meaning that the argument π is correct or not.

We require that the following three properties hold:

- **Completeness:** For any $(x, y) \in R$, if $CRS \leftarrow \text{Setup}(\lambda)$, $\pi \leftarrow \mathcal{P}_{CRS}(x, y)$, then $\mathcal{V}(y, \pi)=1$.
- **Soundness:** For all PPT adversary $\mathcal{A}$,

$$Pr\left[\begin{array}{l}\mathcal{V}(y,\pi^*)=1\\ y\notin L_R\end{array} : \begin{array}{l}CRS \leftarrow Setup(\lambda)\\ (y,\pi^*)\leftarrow\mathcal{A}(CRS)\end{array}\right] \leq negl(\lambda)$$

- **Zero Knowledge:** There exists an efficient simulator $\mathcal{S}=(\mathcal{S}_1, \mathcal{S}_2)$ such that for all PPT adversaries $\mathcal{A}$, $|\Pr[\text{Expt}_{\mathcal{A}}(\lambda)=1]\text{-}\Pr[\text{Expt}^{\mathcal{S}}_{\mathcal{A}}(\lambda)=1| \leq negl(\lambda)$, where the experiments $\text{Expt}_{\mathcal{A}}(\lambda)$ and $\text{Expt}^{\mathcal{S}}_{\mathcal{A}}(\lambda)$ are defined as follows:

$\text{Expt}_{\mathcal{A}}(\lambda)$:	$\text{Expt}^{\mathcal{S}}_{\mathcal{A}}(\lambda)$:
$\text{CRS} \leftarrow \text{Setup}(\lambda)$	$(\text{CRS},\text{AUX}) \leftarrow \mathcal{S}_1(\lambda)$
$\text{return}\mathcal{A}^{\mathcal{P}_{CRS}(\cdot,\cdot)}(\text{CRS})$	$\text{return}\mathcal{A}^{\mathcal{S}'_{\text{CRS}}(\cdot,\cdot,\text{AUX})}(\text{CRS})$

where $\mathcal{S}'_{CRS}(x, y, AUX)=\mathcal{S}_{2_{CRS}}(y, AUX)$.

D. DIFFERING-INPUTS OBFUSCATORS FOR CIRCUITS

The notion of differing-inputs obfuscation was proposed by Barak *et al.* [42] stating that for any two circuits C_0, C_1 with negligible probability that there exists input x on which $C_0(x) \neq C_1(x)$, it should also be hard to distinguish the obfuscation of C_0 from that of C_1.

Let $\{\mathcal{C}_\lambda\}$ be a *differing-inputs circuit family* associated with a PPT Sampler which satisfies that for every PPT adversary $\mathcal{A}$ there exists a negligible function $\epsilon(\lambda)$ such that:

$$Pr\left[\mathcal{C}_0(x)\neq\mathcal{C}_1(x) : \begin{array}{l}(\mathcal{C}_0,\mathcal{C}_1,aux)\leftarrow Sampler(\lambda),\\ x\leftarrow\mathcal{A}(\lambda,\mathcal{C}_0,\mathcal{C}_1,aux)\end{array}\right]\leq\epsilon(\lambda)$$

A uniform PPT algorithm $di\mathcal{O}$ is called a differing-inputs obfuscator for differing-inputs circuit family $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:

- *Correctness*: For all security parameters $\lambda \in \mathbb{N}$, for all $\mathcal{C} \in \{\mathcal{C}_\lambda\}$, for all input x, we have

$$Pr[\mathcal{C}'(x)=\mathcal{C}(x) : \mathcal{C}' \leftarrow di\mathcal{O}(\lambda,\mathcal{C})]=1.$$

- *Polynomial slowdown*: There exists a universal polynomial p such that for any circuit $\mathcal{C}$, we have

$$\mathcal{C}' \mid \leq p(\mid \mathcal{C} \mid), \text{where}\mathcal{C}' = di\mathcal{O}(\lambda,\mathcal{C}).$$

- *Differing-inputs*: For any (not necessarily uniform) PPT algorithms $\mathcal{D}$, there exists a negligible function $\epsilon(\lambda)$ such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for $(\mathcal{C}_0, \mathcal{C}_1, aux) \leftarrow \text{Sampler}(\lambda)$, we have

$$\begin{aligned}&|Pr[\mathcal{D}(di\mathcal{O}(\lambda,\mathcal{C}_0),aux)=1]\\ &\quad -Pr[\mathcal{D}(di\mathcal{O}(\lambda,\mathcal{C}_1),aux)=1]\mid\leq\epsilon(\lambda)\end{aligned}$$

III. PUBLIC-KEY ENCRYPTION WITH KEYWORD SEARCH VIA DIFFERING-INPUTS OBFUSCATION

Technique Outline. In public-key encryption with keyword search, if the tester is allowed to decrypt the keyword ciphertext, which is encrypted by the receiver's public key, the keyword search can easily be achieved by the "decrypt-then-compare" approach. In this section, we adopt this approach by using the power of *differing-inputs obfuscation*, i.e., we set the keyword ciphertext as the ciphertext of the keyword encrypted by a standard public-key encryption scheme under the receiver's public key and set the trapdoor as the obfuscated "decrypt-then-compare" circuit. We hardwire the receiver's secret key and the queried keyword in this circuit. To make the trapdoors for the same keyword indistinguishable, we can hardwire a randomness coin in this circuit in addition.

A. PUBLIC-KEY ENCRYPTION WITH KEYWORD SEARCH: THE MODEL

A PEKS scheme [21] consists of four PPT algorithms (Setup, PEKS, Trapdoor, Test) as follows:

- PEKS.Setup(λ): It takes as input a security parameter λ and outputs a public/secret key pair (pk, sk) for a user.
- PEKS.PEKS (pk,w): It takes as input a public key pk and a keyword w, and outputs a PEKS ciphertext CT of w.
- PEKS.Trapdoor(sk,w'): It takes as input a secret key sk and a keyword w', and outputs a trapdoor $T_{w'}$ of w'.
- PEKS.Test(pk,CT,$T_{w'}$): It takes as input a public key pk, a PEKS ciphertext $CT \leftarrow \text{PEKS}(pk, w)$ and a trapdoor $T_{w'}$, outputs 1 if $w = w'$ and 0 otherwise.

The IND-PEKS-CPA security for PEKS schemes is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- Setup: Taking as input a security parameter λ, the challenger $\mathcal{C}$ runs $(pk, sk) \leftarrow \text{PEKS.Setup}(\lambda)$ and sends the public key pk to $\mathcal{A}$.
- Phase 1: Upon receiving the *Trapdoor* queries for the keyword w from $\mathcal{A}$, $\mathcal{C}$ responds with $T_w \leftarrow \text{PEKS.Trapdoor}(sk, w)$.
- Challenge: When $\mathcal{A}$ sends $\mathcal{C}$ two keywords w_0, w_1 where $|w_0| = |w_1|$ with the restriction that they had not been asked for trapdoors in Phase 1, $\mathcal{C}$ picks a random bit b and sends $c^* \leftarrow \text{PEKS.PEKS}(pk, w_b)$ to $\mathcal{A}$ as the challenge ciphertext.
- Phase 2: $\mathcal{A}$ can continue to issue *Trapdoor* queries for any keyword w with the restriction that $w \neq w_0, w_1$. $\mathcal{C}$ responds the same way as in Phase 1.
- Guess: Finally, the adversary $\mathcal{A}$ outputs a guess b' and succeeds if $b' = b$.

The advantage of the adversary $\mathcal{A}$ in this game is defined as $\text{Adv}^{IND-CPA}_{\mathcal{A},PEKS}(\lambda) = \mid Pr[b = b'] - \frac{1}{2} \mid$.

Definition 1: A PEKS scheme is IND-PEKS-CPA secure if all polynomial-time adversaries $\mathcal{A}$ have at most a negligible advantage in the above game.

B. THE BASIC SCHEME

Let PKE=(KeyGen,Enc,Dec) be an IND-CPA secure public-key encryption scheme and $(\mathcal{P}, \mathcal{V})$ be an adaptively-secure non-interactive zero-knowledge proof system. Our PEKS scheme (Setup, PEKS, Trapdoor, Test) is constructed as follows:

- PEKS.Setup(λ): It runs $(pk_1, sk_1) \leftarrow$PKE.KeyGen(λ), $(pk_2, sk_2) \leftarrow$PKE.KeyGen(λ), and sets user's public/secret key pair as $((pk_1, pk_2, r \leftarrow \{0,1\}^{poly(\lambda)}), sk_1)$.
- PEKS.PEKS (pk,w): Let w be the keyword included in a data file. It computes the PEKS ciphertext CT of w as $CT = (C_1, C_2, \Pi)$, where $C_1 \leftarrow$PKE.Enc($pk_1, w; r_1$), $C_2 \leftarrow$PKE.Enc($pk_2, w; r_2$), $\Pi \leftarrow \mathcal{P}(r, (C_1, C_2), (w, r_1, r_2))$, and r_1, r_2 are the random coins used by PKE.Enc algorithm.
- PEKS.Trapdoor(sk,w'): It generates a searching circuit $\mathcal{P}_{search}$ which is demonstrated in **Algorithm 1**. Then $\mathcal{P}_{search}$ is securely obfuscated to a differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{P}_{search})$. The trapdoor $T_{w'}$ is set as $di\mathcal{O}(\mathcal{P}_{search})$.
- PEKS.Test(pk,CT,$T_{w'}$): It puts the keyword ciphertext CT into trapdoor $T_{w'}$, i.e. the differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{P}_{search})$, and returns the output of $di\mathcal{O}(\mathcal{P}_{search})$.

Algorithm 1 Search Circuit $\mathcal{P}_{search}$

Constant: Receiver's secret key sk_1, queried keyword w'
Input: Keyword ciphertext CT,
Sender's public key $(pk_{s_1}, pk_{s_2}, r_s)$.
Output:

if $\mathcal{V}(r_s, (CT.C_1, CT.C_2), CT.\Pi) == 0$ **then**
  return 0
else
  $w \leftarrow$PKE.Dec($sk_1, CT.C_1$);
  if $w == w'$ **then**
    return 1;
  else
    return 0;

C. SECURITY ANALYSIS

Theorem 1. The above scheme is IND-PEKS-CPA secure.

Proof. We prove the security through a sequence of indistinguishable games.

Game$_0$: is the IND-PEKS-CPA security game of PEKS.

Game$_1$: is the same as Game$_0$, except that upon receiving the *Trapdoor* queries for a keyword, the challenger generates a searching circuit $\mathcal{P}'_{search}$ which is demonstrated in **Algorithm 2**. Then the challenger securely obfuscates it to get the differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{P}'_{search})$ and responds the query with $di\mathcal{O}(\mathcal{P}'_{search})$.

Game$_2$: is the same as Game$_1$, except that instead of $(\mathcal{P}, \mathcal{V})$, its simulator is used to provide the proof. We denote the simulated proof in this game as Π'.

Algorithm 2 Search Circuit $\mathcal{P}'_{search}$

Constant: Receiver's secret key sk_2, queried keyword w'
Input: Keyword ciphertext CT,
Sender's public key $(pk_{s_1}, pk_{s_2}, r_s)$.
Output:

if $\mathcal{V}(r_s, (CT.C_1, CT.C_2), CT.\Pi) == 0$ **then**
  return 0
else
  $w \leftarrow$PKE.Dec($sk_2, CT.C_2$);
  if $w == w'$ **then**
    return 1;
  else
    return 0;

Observe that $\mathcal{P}_{search}$ and $\mathcal{P}'_{search}$ have the same functionality. Thus from the security of differing-inputs obfuscator, $di\mathcal{O}(\mathcal{P}_{search})$ and $di\mathcal{O}(\mathcal{P}'_{search})$ are computationally indistinguishable. Therefore, Game$_1$ is indistinguishable from Game$_0$.

Game$_1$ is indistinguishable from Game$_2$ as if there exists an adversary $\mathcal{A}$ to distinguish Game$_1$ from Game$_2$, then it can be used by an adversary $\mathcal{A}'$ of NIZK to distinguish real proofs from simulated proofs.

Next, we prove the PEKS adversary's advantage in Game$_2$ is negligible by constructing an adversary against the *IND-CPA-secure* PKE scheme. Suppose $\mathcal{A}$ is an adversary of Game$_2$, we can construct an adversary $\mathcal{B}$ against the *IND-CPA-secure* PKE scheme. $\mathcal{B}$ interacts with its own challenge $\mathcal{C}$ in *IND-CPA-secure* game and plays the role of the challenger for $\mathcal{A}$ in Game$_2$ as below:

- Setup: Taking as input a security parameter λ, the challenger $\mathcal{C}$ runs the $(pk_1, sk_1) \leftarrow$PKE.KeyGen(λ) algorithm and sends the public key pk_1 to $\mathcal{B}$. Then B runs $(pk_2, sk_2) \leftarrow$PKE.KeyGen(λ), $r \leftarrow$Sim$_1(\lambda)$ and sends (pk_1, pk_2, r) to $\mathcal{A}$ as PEKS public key. The challenger $\mathcal{C}$ retains the secret key sk_1.
- Phase 1: Upon receiving the *Trapdoor* queries for the keyword w' from $\mathcal{A}$, $\mathcal{B}$ generates a searching circuit $\mathcal{P}'_{search}$ which is demonstrated in **Algorithm 2**.[2] Then $\mathcal{B}$ securely obfuscates it to get the differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{P}'_{search})$. It forwards $di\mathcal{O}(\mathcal{P}'_{search})$ as trapdoor PEKS.$T_{w'}$ to $\mathcal{A}$.
- Challenge: When $\mathcal{A}$ sends $\mathcal{B}$ two keywords w_0, w_1 where $|w_0| = |w_1|$, $\mathcal{B}$ forwards (w_0, w_1) to $\mathcal{C}$. When $\mathcal{B}$ receives its challenge ciphertext $C \leftarrow$PKE.Enc(pk_1, w_β), it computes $C_2 \leftarrow$PKE.Enc($pk_2, w_0; r_0$), $\Pi' \leftarrow$Sim$_2$ (C_1, C_2), and sends $CT = (C, C_2, \Pi')$ to $\mathcal{A}$ as challenge ciphertext of PEKS.
- Phase 2: $\mathcal{A}$ can continue to issue *Trapdoor* queries for any keyword w' with the restriction that $w' \neq w_0, w_1$.
- Guess: Finally, the adversary $\mathcal{B}$ outputs what the adversary $\mathcal{A}$ outputs.

[2] $\mathcal{B}$ knows sk_2

We can get that if $\beta = 0$ with the probability of $\frac{1}{2}$ then $\mathcal{B}$ provides a perfect simulation for $\mathcal{A}$. Otherwise, by the *zero-knowledge property* and *soundness* of the non-interactive zero-knowledge proof system, $\mathcal{A}$ will have a negligible advantage in distinguishing the PEKS ciphertext. Thus, $\mathcal{B}$'s advantage against *IND-CPA-secure* PKE scheme equals to $\frac{1}{2}$ by the advantage of adversary $\mathcal{A}$ in Game_2. Therefore, the advantage of adversary $\mathcal{A}$ in Game_2 is negligible as the based PKE scheme is *IND-CPA-secure*. As Game_2 is indistinguishable from Game_0, the advantage of the adversary in the *IND-PEKS-CPA* security game (Game_0) is negligible.

A note on the leakage-resilience. Commonly, the cryptosystems are implemented in the untrusted environment and it is not assumed that cryptographic keys will be securely kept [54]–[56]. The *leakage-resilient* security considers a kind of side-channel attacks [57]–[59] called *key-leakage attacks* which can recover a fraction of the secret key to the adversary and provides a theoretical method to design cryptosystems proved secure under *key-leakage attacks* [60], [61]. Focusing on a PEKS scheme, the trapdoor leakage was considered in [62], while constructions secure against the continual secret key leakage occurred in the *Trapdoor* process were proposed by Hu *et al.* [63], [64]. In our basic PEKS scheme above, the trapdoor is generated by obfuscating $\mathcal{P}_{search}$ program which only uses the secret key in a decryption implementation of the public-key encryption PKE. Therefore, if the public-key encryption PKE used in $\mathcal{P}_{search}$ is leakage-resilient, our basic PEKS scheme is leakage-resilient.

D. EFFICIENCY IMPROVEMENT AND SECURITY AGAINST KEYWORD GUESSING ATTACKS

In the above construction, the trapdoor is generated as the differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{P}_{search})$ which is inefficient. To improve the efficiency, we slightly change the implementation model of PEKS. Firstly, the receiver generates a searching circuit $\mathcal{C}_{search_{KGA}}$ demonstrated in **Algorithm 3**. Then the receiver securely obfuscates it to get the differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ and sends $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ to the tester only once. When a receiver wants to search in the data, he generates a trapdoor for a keyword w' as $T_{w'} = (T_{w'_1}, T_{w'_2})$=(PKE.Enc$(pk_1, w'||sig_{w'}; r_1)$, PKE.Enc$(pk_2, w'||sig_{w'}; r_2)$) and sends it to the tester, where r_1, r_2 are the random coins used by PKE.Enc algorithm, and $sig_{w'}$ is the signature of w' under the receiver's signing key to guarantee that only the receiver can generate valid trapdoors. The tester executes $di\mathcal{O}(\mathcal{C}_{search})$ taking keyword ciphertext CT and trapdoor $T_{w'}$ as inputs and outputs the search result.

To make our scheme secure against keyword guessing attacks, we also add a signature of keyword w under the sender's signing key in the keyword ciphertext which ensures that the tester cannot generate valid keyword ciphertext of a guessing keyword to test a trapdoor.

Let PKE=(KeyGen,Enc,Dec) be an IND-CPA secure public-key encryption scheme, $(\mathcal{P}, \mathcal{V})$ be an adaptively-secure non-interactive zero-knowledge proof system, and SIG=(KeyGen,Sig,Ver) be a signature scheme achieving Existential UnForgeability against Chosen Message Attack (EUF-CMA secure). Our efficient PEKS scheme (Setup, PEKS, Trapdoor, Test) secure against keyword guessing attacks is constructed as follows:

- PEKS.Setup(λ): It runs $(pk_1, sk_1) \leftarrow$PKE.KeyGen(λ), $(pk_2, sk_2) \leftarrow$PKE.KeyGen(λ), $(pk_3, sk_3) \leftarrow$ SIG.KeyGen(λ), and sets user's public/secret key pair as $((pk_1, pk_2, pk_3, r \leftarrow \{0,1\}^{poly(\lambda)}), (sk_1, sk_3))$.
- PEKS.PEKS (pk,w): Let w be the keyword included in a data file. Let $((pk_{s_1}, pk_{s_2}, pk_{s_3}, r_s \leftarrow \{0,1\}^{poly(\lambda)}), (sk_{s_1}, sk_{s_3}))$ be the sender's public/secret key pair. Let $((pk_1, pk_2, pk_3, r \leftarrow \{0,1\}^{poly(\lambda)}), (sk_1, sk_3))$ be the receiver's public/secret key pair. It obtains a signature sig_w of w under the sender's signing key sk_{s_3} and computes the PEKS ciphertext CT of w as $CT = (C_1, C_2, \Pi)$, where $C_1 \leftarrow$PKE.Enc$(pk_1, w||sig_w; r_1)$, $C_2 \leftarrow$PKE.Enc$(pk_2, w||sig_w; r_2)$, $\Pi \leftarrow \mathcal{P}(r, (C_1, C_2, (w, r_1, r_2))$, and r_1, r_2 are the random coins used by PKE.Enc algorithm.
- PEKS.Trapdoor(sk,w'): The receiver first generates a searching circuit $\mathcal{C}_{search_{KGA}}$ which is demonstrated in **Algorithm 3**. Then $\mathcal{C}_{search_{KGA}}$ is securely obfuscated to a differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$. It sends $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ to the tester only once. In the following, to search files with keyword w', the receiver sets the trapdoor for w' as $T_{w'}$=$(T_{w'_1}, T_{w'_2})$=(PKE.Enc$(pk_1, w'||sig_{w'}; r_1)$, PKE.Enc$(pk_2, w'||sig_{w'}; r_2)$, where $sig_{w'}$ is the signature of w' under the receiver's signing key sk_3 and r_1, r_2 are the random coins used by PKE.Enc algorithm. Then the receiver sends $T_{w'}$ to the tester.
- PEKS.Test(pk,CT,$T_{w'}$): The tester executes $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ taking keyword ciphertext CT and trapdoor $T_{w'}$ as inputs, and outputs the search result.

Theorem 2. The above scheme is IND-PEKS-CPA secure.

Proof. The *IND-PEKS-CPA* security of the improved scheme can be proved similar to that of our basic PEKS scheme. We also need a sequence of games as follows:

Game_0: is the IND-PEKS-CPA security game of PEKS. Note that in this game, the challenger only generates the obfuscated circuit $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ once and can send it to the adversary in Setup phase.

Game_1: is the same as Game_0, except that the challenger generates a obfuscated searching circuit $di\mathcal{O}(\mathcal{C}'_{search_{KGA}})$ instead of $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ where $\mathcal{C}'_{search_{KGA}}$ is demonstrated in **Algorithm 4**.

Game_2: is the same as Game_1, except that instead of $(\mathcal{P}, \mathcal{V})$, its simulator is used to provide the proof. We denote the simulated proof in this game as Π'.

Observe that $\mathcal{C}_{search_{KGA}}$ and $\mathcal{C}'_{search_{KGA}}$ have the same functionality. Thus from the security of differing-inputs obfuscator, $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ and $di\mathcal{O}(\mathcal{C}'_{search_{KGA}})$ are computationally indistinguishable. Therefore, Game_1 is indistinguishable from Game_0.

Algorithm 3 Search Circuit $\mathcal{C}_{search_{KGA}}$

Constant: Receiver's secret key sk_1,
receiver's public key (pk_1, pk_2, pk_3, r),
sender's public key $(pk_{s_1}, pk_{s_2}, pk_{s_3}, r_s)$.
Input: Keyword ciphertext CT, trapdoor $T_{w'}$.
Output:

if $\mathcal{V}(r_s, (CT.C_1, CT.C_2), CT.\Pi) == 0$ **then**
  return 0
else
  $w_1||sig_{w_1} \leftarrow$PKE.Dec$(sk_1, CT.C_1)$;
  if SIG.Ver$(pk_{s_3}, sig_{w_1}, w_1)$==false **then**
    return 0;
  else
    $w_2||sig_{w_2} \leftarrow$PKE.Dec$(sk_1, T_{w'}.T_{w'_1})$;
    if SIG.Ver(pk_3, sig_{w_2}, w_2)==false **then**
      return 0;
    else
      if $w_1 == w_2$ **then**
        return 1;
      else
        return 0;

Algorithm 4 Search Circuit $\mathcal{C}'_{search_{KGA}}$

Constant: Secret key sk_2,
receiver's public key (pk_1, pk_2, pk_3, r),
sender's public key $(pk_{s_1}, pk_{s_2}, pk_{s_3}, r_s)$.
Input: Keyword ciphertext CT, trapdoor $T_{w'}$.
Output:

if $\mathcal{V}(r_s, (CT.C_1, CT.C_2), CT.\Pi) == 0$ **then**
  return 0
else
  $w_1||sig_{w_1} \leftarrow$PKE.Dec$(sk_2, CT.C_2)$;
  if SIG.Ver$(pk_{s_3}, sig_{w_1}, w_1)$==false **then**
    return 0;
  else
    $w_2||sig_{w_2} \leftarrow$PKE.Dec$(sk_2, CT.C_2)$;
    if SIG.Ver(pk_3, sig_{w_2}, w_2)==false **then**
      return 0;
    else
      if $w_1 == w_2$ **then**
        return 1;
      else
        return 0;

Game$_1$ is indistinguishable from Game$_2$ as if there exists an adversary $\mathcal{A}$ to distinguish Game$_1$ from Game$_2$, then it can be used by an adversary $\mathcal{A}'$ of NIZK to distinguish real proofs from simulated proofs.

Next, we prove the PEKS adversary's advantage in Game$_2$ is negligible by constructing an adversary against the *IND-CPA-secure* PKE scheme. Suppose $\mathcal{A}$ is an adversary of Game$_2$, we can construct an adversary $\mathcal{B}$ against the *IND-CPA-secure* PKE scheme. $\mathcal{B}$ interacts with its own challenge $\mathcal{C}$ in *IND-CPA-secure* game and plays the role of a challenger for $\mathcal{A}$ in Game$_2$ as below:

- Setup: Taking as input a security parameter λ, the challenger $\mathcal{C}$ runs the $(pk_1, sk_1) \leftarrow$PKE.KeyGen(λ) algorithm and sends the public key pk_1 to $\mathcal{B}$. Then $\mathcal{B}$ runs $(pk_2, sk_2) \leftarrow PKE.KeyGen(\lambda)$; $(pk_3, sk_3) \leftarrow SIG.KeyGen(\lambda)$; $r \leftarrow Sim_1(\lambda)$; and sends (pk_1, pk_2, pk_3, r) to $\mathcal{A}$ as PEKS public key. The challenger $\mathcal{C}$ retains the secret key sk_1. $\mathcal{B}$ also generates the obfuscated circuit $di\mathcal{O}(\mathcal{C}'_{search})$ and forwards it to $\mathcal{A}$.
- Phase 1: Upon receiving the *Trapdoor* queries for the keyword w' from $\mathcal{A}$, $\mathcal{B}$ generates $T_{w'}$ and sends it to $\mathcal{A}$.
- Challenge: When $\mathcal{A}$ sends $\mathcal{B}$ two keywords w_0, w_1 where $|w_0| = |w_1|$, $\mathcal{B}$ obtains the signatures of w_0, w_1 as sig_{w_0}, sig_{w_1} respectively under sk_3 and forwards $(w_0||sig_{w_0}, w_1||sig_{w_1})$ as its challenge plaintext to $\mathcal{C}$. When $\mathcal{B}$ receives its challenge ciphertext $C \leftarrow$PKE.Enc$(pk_1, w_\beta||sig_{w_\beta})$, it computes

$$C_2 \leftarrow PKE.Enc(pk_2, w_0||sig_{w_0}; r_0),$$
$$\Pi' \leftarrow Sim_2(C_1, C_2),$$

  and sends $CT = (C, C_2, \Pi')$ to $\mathcal{A}$ as challenge ciphertext of PEKS.
- Phase 2: $\mathcal{A}$ can continue to issue *Trapdoor* queries for any keyword w' with the restriction that $w' \neq w_0, w_1$.
- Guess: Finally, the adversary $\mathcal{B}$ outputs what the adversary $\mathcal{A}$ outputs.

We can get that if $b = 0$ with the probability of $\frac{1}{2}$ then $\mathcal{B}$ provides a perfect simulation for $\mathcal{A}$. Otherwise, by the *zero-knowledge property* and *soundness* of non-interactive zero-knowledge proof system, $\mathcal{A}$ will have a negligible advantage in distinguishing the PEKS ciphertext. Thus, $\mathcal{B}$'s advantage against *IND-CPA-secure* PKE scheme equals to $\frac{1}{2}$ by the advantage of adversary in Game$_2$. Therefore, the advantage of adversary in Game$_2$ is negligible as the based PKE scheme is *IND-CPA-secure*. As Game$_2$ is indistinguishable from Game$_0$, advantage of adversary in the *IND-PEKS-CPA* security game (Game$_0$) is negligible.

Theorem 3. The above scheme is secure against off-line keyword guessing attacks.

Proof. The scheme is secure under keyword guessing attacks as if a KGA adversary can forge a PEKS ciphertext of keyword $\bar{w}$ which can pass the test of a trapdoor $T_{w'}$ then a valid signature of SIG can be forged. We prove the KGA adversary's advantage is negligible by constructing an adversary against the *EUF-CMA* secure signature scheme. Suppose $\mathcal{A}$ is an adversary implementing keyword guessing attacks, we can construct an adversary $\mathcal{B}$ against the *EUF-CMA* secure signature scheme. $\mathcal{B}$ interacts with its own challenge $\mathcal{C}$ in *EUF-CMA* game and interacts with $\mathcal{A}$ as below:

- Setup: Taking as input a security parameter λ, the challenger $\mathcal{C}$ runs the $(pk_{s_3}, sk_{s_3}) \leftarrow$SIG.KeyGen($\lambda$) algorithm and sends the public key pk_{s_3} to $\mathcal{B}$. Then $\mathcal{B}$ runs

$$(pk_1, sk_1) \leftarrow PKE.KeyGen(\lambda);$$

TABLE 1. Comparison with existing PEKS schemes resisting KGA.

Scheme	Computation		Communication in a search	No Designated Tester	Against Malicious User	Against Malicious Server	Functionality
	Trapdoor	Test					
[31]	$O(1)$	$O(n)$	$O(r)$	×	✓	×	Single
[33]	$O(1)$	$O(n)$	$O(r)$	×	✓	×	Single
[32]	$O(1)$	$O(n)$	$O(r)$	×	✓	×	Single
[34]	$O(1)$	$O(n)$	$O(r)$	×	✓	×	Single
Our Scheme	$O(1)$	$O(n)$	$O(r)$	✓	✓	✓	Can be easily extended

- n denotes the number of total files and r denotes the number of the retrieving files for a certain keyword.
- This is a theoretical analysis for the computation overheads. Actually, our obfuscation-based scheme is less efficient than other works in practical scenario.

$$(pk_2, sk_2) \leftarrow PKE.KeyGen(\lambda);$$
$$(pk_3, sk_3) \leftarrow SIG.KeyGen(\lambda);$$
$$r \leftarrow Sim_1(\lambda);$$

and sends (pk_1, pk_2, pk_3, r) to $\mathcal{A}$ as the receiver's public key. Also, $\mathcal{B}$ runs

$$(pk_{s_1}, sk_{s_1}) \leftarrow \text{PKE.KeyGen}(\lambda);$$
$$(pk_{s_2}, sk_{s_2}) \leftarrow \text{PKE.KeyGen}(\lambda);$$
$$r_s \leftarrow \text{Sim}_1(\lambda);$$

and sends $(pk_{s_1}, pk_{s_2}, pk_{s_3}, r_s)$ to $\mathcal{A}$ as the sender's public key. The challenger $\mathcal{C}$ retains the secret key sk_{s_3}. $\mathcal{B}$ also generates the obfuscated circuit $di\mathcal{O}(\mathcal{C}'_{searchKGA})$ and forwards it to $\mathcal{A}$.

- Challenge: $\mathcal{B}$ chooses a keyword w', and generates the trapdoor $T_{w'}$ for it. $\mathcal{B}$ sends $T_{w'}$ to $\mathcal{A}$. Note that $\mathcal{B}$ has all the public/secret key pairs of the receiver, it can compute the trapdoor.
- Guess: Finally, $\mathcal{A}$ chooses a keyword $\bar{w}$. $\mathcal{A}$ computes and sends to $\mathcal{B}$ the keyword ciphertext $CT_{\bar{w}}$ of $\bar{w}$. If Test($CT_{\bar{w}}$,$T_{w'}$)=1, $\mathcal{B}$ parses PKE.Dec(sk_1, $CT_{\bar{w}}.C_1$) as $\bar{w}||sig_{\bar{w}}$, and outputs ($\bar{w}$, $sig_{\bar{w}}$) as a forged valid signature of $\bar{w}$.

We can observe that if $\mathcal{A}$ can break the keyword guessing attack security of the PEKS scheme, $\mathcal{B}$ can forge a valid signature of SIG. As the signature scheme, SIG is EUF-CMA secure, the advantage of the adversary in keyword guessing attack security game is negligible. Note that compared with other PEKS schemes secure against off-line keyword guessing attacks, our scheme is a standard PEKS scheme allowing anyone to execute the tester process as long as he has the PEKS ciphertexts and trapdoors, while in other schemes, only a designated tester can respond to search queries.

E. PERFORMANCE AND FUNCTIONALITIES

Our scheme can be easily extended to schemes which support other functionalities, such as multiple keywords search, fuzzy keyword search, etc. For example, if we want to enable the scheme with multiple keywords search, we can set the searched keyword as $w_1||w_2||\cdots||w_n$, where w_is are multiple keywords associated with PEKS ciphertext/trapdoor. Then, we will only need to slightly modify the codes from Line 12 to 15 in $\mathcal{C}_{searchKGA}$ to a suitable keyword-based search algorithm under plaintexts and its corresponding $di\mathcal{O}$ circuit.

TABLE 2. Benchmark.

Operation	Time
Pairing	$T_{Pairing}$ ≈9.2 ms
Element exponentiation in G	$T_{E,G}$ ≈ 10.56 ms
Element inverse	T_{Inv} ≈0.013 ms
DSA signing	T_{Sig} ≈1.5 ms
RSA-OAEP pub enc	T_{PubEnc} ≈2.4 ms

Next, we will give a performance analysis of our improved theoretical obfuscation-based scheme and compare them with some existing works. In our analysis, we focus on the computation and communication overheads in the trapdoor generation and test operation due to the high frequency. To make successful keyword searches, the receiver needs to generate an obfuscated search program $di\mathcal{O}(\mathcal{C}_{searchKGA})$ once and sends it to the server. Then the receiver only needs one signing and two public key encrypting operations for each queried keyword. On the server side, to test a trapdoor with keyword ciphertexts, it only needs to execute the obfuscated search program $di\mathcal{O}(\mathcal{C}_{searchKGA})$.

Table 1 gives the comparison between our scheme and some other schemes resisting keyword guessing attacks. From the comparison, we can see that our improved scheme can achieve several good properties, e.g. no designated tester, resistance to malicious server and easy functionality extension while keeping the asymptotic complexity unchanged.

For concrete efficiency, note that obfuscated search program is executed by the test operation which is inefficient in real life. Therefore, we can only compare the computational time of the trapdoor generation process between our improved scheme and some other schemes resisting keyword guessing attacks. The efficiency is evaluated on a personal computer with Intel Core i7-4600 CPU, and 8GB of DDR3 RAM. Firstly, we evaluate the trapdoor generation performance of our improved scheme. As there only needs to execute signing once and encrypting under public key twice, we use well-known DSA signature scheme and RSA-OAEP public key encryption scheme as the concrete public key cryptosystems. The DSA and RSA-OAEP schemes are implemented in Python by calling modules of PyCrypto (Python Cryptography Toolkit) library [65], which provides interfaces that can be used to implement cryptographic primitives. Both

TABLE 3. Computational costs of the trapdoor generation.

Scheme	Time
[31]	$3T_{E,G}$+T_{Inv}≈31.693 ms
[32]	$T_{E,G}$+T_{Inv}≈ 10.573 ms
[34]	$T_{E,G}$+$T_{Pairing}$≈19.76 ms
Our scheme	$2T_{PubEnc} + T_{Sig}$ ≈6.3 ms

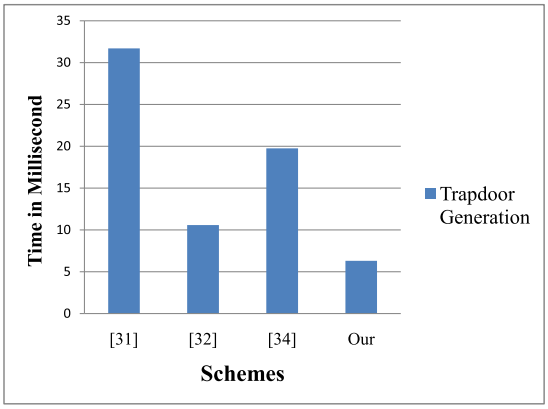


FIGURE 1. Comparison about computational time of trapdoor generation.

of the implemented DSA and RSA-OAEP schemes achieve 1024-bit security level. By the implementation, the trapdoor generation time of our improved scheme is about 6.3 ms. Secondly, we obtain the trapdoor generation performance of some other schemes such as [31], [32], and [34]. As all of these schemes are constructed in prime order bilinear groups, we implement them based on PBC (Pairing-Based Crypto) Library [66]. In order to obtain the runtime of them and get an accurate comparison, we choose the Type A curve $y^2 = x^3 + x$ and 512-bit security parameter length. The trapdoor generation algorithms of these schemes depend on three basic element operations: element exponentiation in G, pairing and element inverse. Therefore, we first test these element operations in prime order bilinear groups and show the benchmarks in Table 2. Then we analyze the total computational costs of the trapdoor generation of these schemes based on the benchmarks. The comparison results between our improved scheme and these schemes [31], [32], [34] are described in Table 3 and Fig. 1.

F. EXTENSION TO THE MULTI-USER SETTING

In the multi-user setting, data owner hopes to share their data with multiple users and permits the users to search over the encrypted shared data in the cloud. The above PEKS scheme can be easily extended to a PEKS scheme in the multi-user setting. The technique behind the construction of PEKS scheme in the multi-user setting is that: the data owner provides the cloud server with a differing-inputs obfuscated circuit $di\mathcal{O}(\mathcal{C}_{re-enc})$ which makes a re-encryption [67] of a PEKS ciphertext destined for user i and turns it into that for user j. Then the cloud server can execute this obfuscated circuit to get the PEKS ciphertext for user j and search on it with trapdoors generated by user j. In the PEKS ciphertext of keyword w, we add a ciphertext of w encrypted by a block cipher algorithm such as AES, under a secret key k which is holden secretly by the sender (data owner). This ciphertext is used in the re-encryption circuit $\mathcal{C}_{re-enc}$ which is described in **Algorithm 5**.

Let PKE=(KeyGen,Enc,Dec) be an IND-CPA secure public-key encryption scheme, $(\mathcal{P}, \mathcal{V})$ be an adaptively-secure non-interactive zero-knowledge proof system, SIG=(KeyGen,Sig,Ver) be an EUF-CMA secure signature scheme and BC=(KeyGen,Enc,Dec) be a block cipher algorithm. We describe our PEKS scheme in the multi-user setting as follows:

- PEKS.Setup(λ): It runs $(pk_1, sk_1) \leftarrow$PKE.KeyGen(λ), $(pk_2, sk_2) \leftarrow$PKE.KeyGen(λ), $(pk_3, sk_3) \leftarrow$ SIG.KeyGen(λ), and sets user's public/secret key pair as $((pk_1, pk_2, pk_3, r \leftarrow \{0,1\}^{poly(\lambda)}), (sk_1, sk_3))$.
- PEKS.PEKS (pk,w): Let w be the keyword associated to a data file. Let $((pk_{s_1}, pk_{s_2}, pk_{s_3}, r_s \leftarrow \{0,1\}^{poly(\lambda)}), (sk_{s_1}, sk_{s_3}))$ be the sender's public/secret key pair. Let $((pk_1, pk_2, pk_3, r \leftarrow \{0,1\}^{poly(\lambda)}),(sk_1, sk_3))$ be the receiver's public/secret key pair. The sender generates a secret key k of a block cipher and holds it secretly. The sender obtains a signature sig_w of w under the sender's signing key sk_{s_3} and computes the PEKS ciphertext CT of w as $CT = (C_1, C_2, \Pi, C_3)$, where

$$\begin{aligned} C_1 &\leftarrow PKE.Enc(pk_1, w||sig_w; r_1), \\ C_2 &\leftarrow PKE.Enc(pk_2, w||sig_w; r_2), \\ \Pi &\leftarrow \mathcal{P}(r, (C_1, C_2), (w, r_1, r_2)), \\ C_3 &\leftarrow BC.Enc(k, w||sig_w), \end{aligned}$$

  and r_1, r_2 are the random coins used by PKE.Enc algorithm.
- PEKS.Trapdoor(sk,w'): The receiver first generates a searching circuit $\mathcal{C}_{search_{KGA}}$ which is demonstrated in **Algorithm 3**. Then $\mathcal{C}_{search_{KGA}}$ is securely obfuscated to a differing-inputs obfuscation circuit $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$. It sends $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ to the tester only once. In the following, to search files containing keyword w', the receiver sets the trapdoor for w' as $T_{w'}$=$(T_{w'_1}, T_{w'_2})$ where

$$\begin{aligned} T_{w'_1} &\leftarrow PKE.Enc(pk_1, w'||sig_{w'}; r_1), \\ T_{w'_2} &\leftarrow PKE.Enc(pk_2, w'||sig_{w'}; r_2), \end{aligned}$$

  and $sig_{w'}$ is the signature of w' under the receiver's signing key sk_3 and r_1, r_2 are the random coins used by PKE.Enc algorithm. Then the receiver sends $T_{w'}$ to the tester.
- PEKS.Test(pk,CT,$T_{w'}$): The tester executes $di\mathcal{O}(\mathcal{C}_{search_{KGA}})$ taking keyword ciphertext CT and trapdoor $T_{w'}$ as inputs, and outputs the search result.
- PEKS.Re-encryption(CT_i,pk_j): The sender provides the tester with an indistinguishability obfuscated circuit $di\mathcal{O}(\mathcal{C}_{re-enc})$ of $\mathcal{C}_{re-enc}$ described in **Algorithm 5**. The

TABLE 4. Comparison with existing PEKS scheme in the multi-user setting.

Scheme	Ciphertext length	KGA security	Dynamic users	No Trust Third Party
[38]	linear	×	×	✓
[35]	constant	×	✓	×
[41]	constant	×	✓	✓
[39]	linear	×	×	✓
[36]	constant	×	×	×
[37]	constant	×	✓	×
[40]	constant	×	✓	✓
Our Scheme	constant	✓	✓	✓

- *linear* means the ciphertext length is linear with the number of all the users.
- *Dynamic users* means that it does not needs to pre-fix all the users. The user can generate his own public/secret key pair by himself.
- The overheads of [38] and [39] is for single keyword search.
- In [41], the data owner plays a role of the trust third party.
- [40] gets rid of the trust third party by implementing a complex agreement process among all users.

tester executes this circuit taking as inputs the keyword ciphertext CT_i for user i, the public key pk_j of user j, and outputs the keyword ciphertext CT_j for user j.

Algorithm 5 Re-encryption Circuit $\mathcal{C}_{re-enc}$

Constant: A secret key k of the block cipher scheme holden secretly by the sender, a random key k_1 used to generate randomness by the pseudorandom function.

Input: Keyword ciphertext CT_i for user i, the public key $(pk_{j_1}, pk_{j_2}, pk_{j_3}, r_j)$ of user j.

Output:

$w||sig_w \leftarrow \text{BC.Dec}(k, CT_i.C_3)$;
$r'_1 \leftarrow \text{PRF}(k_1, CT_i.C_1)$;
$CT_j.C_1 \leftarrow \text{PKE.Enc}(pk_{j_1}, w||sig_w; r'_1)$;
$r'_2 \leftarrow \text{PRF}(k_1, CT_i.C_2)$;
$CT_j.C_2 \leftarrow \text{PKE.Enc}(pk_{j_2}, w||sig_w; r'_2)$;
$CT_j.\Pi \leftarrow \mathcal{P}(r_j, (CT_j.C_1, CT_j.C_2), (w, r_1, r_2))$;
$CT_j.C_3 = CT_i.C_3$;
return CT_j;

Table 4 shows the differences between our scheme and some other schemes in the multi-user setting. Our scheme achieves some good features in the multi-user setting, i.e., constant keyword ciphertext length, no trust third party, supporting dynamic users and resisting keyword guessing attacks.

IV. CONCLUSION

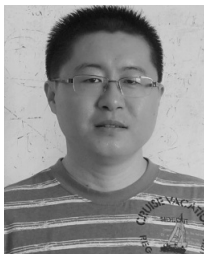
In this paper, we propose public-key encryption with keyword search schemes based on differing-inputs obfuscation as an initial attempt. We present a basic PEKS scheme supporting single-keyword search which can be easily extended to support complex functionalities and to that in the multi-user setting. Moreover, we consider the KGA security of PEKS and improve our basic scheme to resist off-line keyword guessing attacks. Compared with dPEKS schemes resisting KGA attacks, our scheme is a standard one rather than a PEKS scheme with a designated tester. As the constructions are obtained by applying obfuscation, the limitation of our schemes is obvious, i.e., the trapdoor generation process is inefficient which may make them inapplicable in resource-limited environments. In spite of the results obtained by applying obfuscation in PEKS construction, we must point out that our PEKS schemes fail to verify the correctness and completeness of the search result from the server. We leave it an open problem.

REFERENCES

[1] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, "Access-control language for multidomain environments," *IEEE Internet Comput.*, vol. 8, no. 6, pp. 40–50, Nov./Dec. 2004.
[2] H. Zhong, W. Zhu, Y. Xu, and J. Cui, "Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage," *Soft Comput.*, vol. 22, pp. 243–251, Jan. 2018.
[3] M. Blaze *et al.*, "Dynamic trust management," *Computer*, vol. 42, no. 2, pp. 44–52, Feb. 2009.
[4] D. Shin and G.-J. Ahn, "Role-based privilege and trust management," *Comput. Syst. Sci. Eng.*, vol. 20, no. 6, pp. 401–410, Nov. 2005.
[5] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. ACM Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, pp. 598–609, Oct. 2007.
[6] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.
[7] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.
[8] S. Li, J. Cui, H. Zhong, and Q. He, "LEPA: A lightweight and efficient public auditing scheme for cloud-assisted wireless body sensor networks," *Secur. Commun. Netw.*, vol. 2017, no. 11, Jul. 2017, Art. no. 4364376.
[9] R. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proc. ACM Symp. Operating Syst. Princ.* New York, NY, USA: ACM, Oct. 2011, pp. 85–100.
[10] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, and D. S. Wong, "L-EncDB: A lightweight framework for privacy-preserving data queries in cloud computing," *Knowl.-Based Syst.*, vol. 79, pp. 18–26, May 2015.
[11] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searching on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[12] E.-J. Goh, "Secure indexes," *Cryptol. ePrint Arch.*, vol. 2003, p. 216, 2003.
[13] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, Jun. 2004, pp. 31–45.

[14] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. Int. Conf. Inf. Comput. Sci.*, Dec. 2005, pp. 414–426.
[15] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 441–445.
[16] H. Li, Y. Yang, Y. Dai, J. Bai, S. Yu, and Y. Xiang, "Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data," *IEEE Trans. Cloud Comput.*, to be published. doi: 10.1109/TCC.2017.2769645.
[17] H. Li, D. Liu, Y. Dai, T. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 97–109, Jan./Mar. 2018.
[18] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.
[19] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Apr. 2019.
[20] X. Ge *et al.*, "Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification," *IEEE Trans. Dependable Secure Comput.*, to be published. doi: 10.1109/TDSC.2019.2896258.
[21] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2004, pp. 506–522.
[22] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1984, pp. 47–53.
[23] M. Abdalla *et al.*, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Advances Cryptology—CRYPTO*. Berlin, Germany: Springer, 2005, pp. 205–222.
[24] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptogr. Conf.*, Feb. 2007, pp. 535–554.
[25] S. Sedghi, P. Van Liesdonk, S. Nikova, P. Hartel, and W. Jonker, "Searching keywords with wildcards on encrypted data," in *Proc. Int. Conf. Secur. Cryptogr. Netw.*, Sep. 2010, pp. 138–153.
[26] J. Lai, X. Zhou, R. H. Deng, Y. Li, and K. Chen, "Expressive search on encrypted data," in *Proc. ACM Symp. Inf., Comput. Commun. Secur.*, Dec. 2013, pp. 243–252.
[27] D. Koo, J. Hur, and H. Yoon, "Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage," *Comput. Electr. Eng.*, vol. 39, no. 1, pp. 34–46, Jan. 2013.
[28] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "AKSER: Attribute-based keyword search with efficient revocation in cloud computing," *Inf. Sci.*, vol. 423, pp. 343–352, Jan. 2018.
[29] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2006, pp. 75–83.
[30] B. Zhu, B. Zhu, and K. Ren, "Peksrand: Providing predicate privacy in public-key encryption with keyword search," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–6.
[31] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *J. Syst. Softw.*, vol. 83, no. 5, pp. 763–771, 2010.
[32] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Inf. Sci.*, vol. 238, pp. 221–241, Jul. 2013.
[33] Y.-C. Chen, "SPEKS: Secure server-designation public key encryption with keyword search against keyword guessing attacks," *Comput. J.*, vol. 58, no. 4, pp. 922–933, Apr. 2015.
[34] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vols. 403–404, pp. 1–14, Sep. 2017.
[35] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Proc. Annu. IFIP WG Work. Conf. Data Appl. Secur.*, Jul. 2008, pp. 127–143.
[36] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur.*, Sep. 2016, pp. 173–195.
[37] D. Sharma and D. Jinwala, "Multiuser searchable encryption with token freshness verification," *Secur. Commun. Netw.*, vol. 2017, Nov. 2017, Art. no. 6435138.
[38] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pairing-Based Cryptogr.*, Jul. 2007, pp. 2–22.
[39] Z. Lv, C. Hong, M. Zhang, and D. Feng, "Expressive and secure searchable encryption in the public key setting," in *Proc. Int. Conf. Inf. Secur.*, Oct. 2014, pp. 364–376.
[40] H. Wang, X. Dong, and Z. Cao, "Secure and efficient encrypted keyword search for multi-user setting in cloud computing," *Peer-to-Peer Netw. Appl.*, vol. 12, no. 1, pp. 32–42, Jan. 2019.
[41] Q. Tang, "Nothing is for free: Security in searching shared and encrypted data," *IEEE Trans Inf. Forensics Security*, vol. 9, no. 11, pp. 1943–1952, Nov. 2014.
[42] B. Barak *et al.*, "On the (IM) possibility of obfuscating programs," in *Advances Cryptology—CRYPTO*. Berlin, Germany: Springer, 2001, pp. 1–18.
[43] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attacks," in *Proc. ACM Symp. Theory Comput.*, May 1990, pp. 427–437.
[44] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 40–49.
[45] A. Sahai and B. Waters, "How to use indistinguishability obfuscation: Deniable encryption, and more," in *Proc. ACM Symp. Theory Comput.*, May 2014, pp. 475–484.
[46] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren, "Verifiable searchable symmetric encryption from indistinguishability obfuscation," in *Proc. 10th ACM Symp. Inf., Comput. Commun. Secur.*, Oct. 2015, pp. 621–626.
[47] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric-key based proofs of retrievability supporting public verification," in *Proc. Eur. Symp. Res. Comput. Secur.*, Sep. 2015, pp. 203–223.
[48] D. Khurana, V. Rao, and A. Sahai, "Multi-party key exchange for unbounded parties from indistinguishability obfuscation," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2015, pp. 52–75.
[49] D. Boneh and M. Zhandry, "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation," *Algorithmica*, vol. 79, no. 4, pp. 1233–1285, Dec. 2017.
[50] S. Ohata and K. Matsuura, "Proxy re-encryption via indistinguishability obfuscation," *Secur. Commun. Netw.*, vol. 9, pp. 1786–1795, Aug. 2016.
[51] M. Zhang, Y. Jiang, H. Shen, and W. Susilo, "Cloud-based data-sharing scheme using verifiable and cca-secure re-encryption from indistinguishability obfuscation," in *Proc. Int. Conf. Inf. Secur. Cryptologyhttp*, Dec. 2018, pp. 1–20.
[52] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, no. 2, pp. 281–308, 1988.
[53] S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi, "On the non-malleability of the Fiat-Shamir transform," in *Proc. Int. Conf. Cryptol. India*, Dec. 2012, pp. 60–79.
[54] K. Harrison and S. Xu, "Protecting cryptographic keys from memory disclosure attacks," in *Proc. Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2007, pp. 137–143.
[55] A. Juma and Y. Vahlis, "Protecting cryptographic keys against continual leakage," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2010, pp. 41–58.
[56] M. S. Inci, B. Gulmezoglu, G. Irazoqui, T. Eisenbarth, and B. Sunar, "Seriously, get off my cloud! Cross-VM RSA key recovery in a public cloud," *Cryptol. ePrint Arch.*, vol. 2015, p. 898, 2015.
[57] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1996, pp. 104–113.
[58] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology—CRYPTO*. Springer, 1997, pp. 513–525.
[59] J. A. Halderman *et al.*, "Lest we remember: Cold-boot attacks on encryption keys," in *Proc. 17th USENIX Secur. Symp.*, Jul. 2008, pp. 45–60.
[60] A. Akavia, S. Goldwasser, and V. Vaikuntanathan, "Simultaneous hardcore bits and cryptography against memory attacks," in *Proc. Theory Cryptogr. Conf.*, Mar. 2009, pp. 474–495.
[61] M. Naor and G. Segev, "Public-key cryptosystems resilient to key leakage," *SIAM J. Comput.*, vol. 41, no. 4, pp. 772–814, Aug. 2012.

[62] Y. Chen, Z. Zhang, D. Lin, and Z. Cao, "Anonymous identity-based hash proof system and its applications," in *Proc. Int. Conf. Provable Secur.*, Sep. 2012, pp. 143–160.
[63] C. Hu, R. Yang, P. Liu, Z. Yu, Y. Zhou, and Q. Xu, "Public-key encryption with keyword search secure against continual memory attacks," *Secur. Commun. Netw.*, vol. 9, no. 11, pp. 1613–1629, Jul. 2016.
[64] C. Hu, Z. Li, P. Liu, R. Yang, S. Guo, and H. Zhang, "Verifiable public-key encryption with keyword search secure against continual memory attacks," in *Mobile Networks and Applications*. Berlin, Germany: Springer, 2018. doi: 10.1007/s11036-018-1101-4.
[65] D. Litzenberger. (2002). *Python Cryptography Toolkit*. [Online]. Available: https://pypi.org/project/pycrypto/
[66] B. Lynn. (2007). *PBC Library-The Pairing-Based Cryptography Library*. [Online]. Available: http://crypto.stanford.edu/pbc/
[67] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.



CHENGYU HU received the Ph.D. degree from Shandong University, in 2008, where he is currently a Lecturer. His main research interests include cloud system security, public-key cryptography, and leakage-resilient cryptography.

PENGTAO LIU received the master's degree in computer software and theory from Shandong University, in 2006. She is currently with the School of Cyberspace Security, Shandong University of Political Science and Law. Her main research interests include public key encryption and digital signature.

RUPENG YANG received the master's degree from Shandong University, in 2015, where he is currently pursuing the Ph.D. degree. His main research interests include cloud system security, leakage-resilient cryptography, blockchain technology, and lattice-based cryptography.

YAN XU received the B.S. degree from Shandong University, Jinan, China, in 2004, and the Ph.D. degree in computer science and technology from the University of Science and Technology of China (USTC), China, in 2015. She is currently an Associate Professor with Anhui University. Her research interest includes network and information security.

RSA-CRT 密码防御算法的故障注入攻击

孔凡玉[1]，乔咏[2]，刘蓬涛[3]，刘晓东[1]，周大水[1]

（1. 山东大学网络信息安全研究所，山东 济南 250100；
2. 中标软件有限公司，北京 100190；
3. 山东政法学院网络空间安全学院，山东 济南 250014）

摘　要：RSA 密码是在 TLS、SSL、IPSec 等网络安全协议中广泛使用的密码算法，其安全性至关重要。在 FDTC 2014 会议上，Rauzy 和 Guilley 提出了改进的基于中国剩余定理的 RSA 密码实现算法，用于抵抗故障注入攻击。针对 Rauzy 和 Guilley 的两个 RSA-CRT 安全防御算法，提出了相应的故障注入攻击方法，在 RSA 密码运算过程中注入一个永久性错误，并利用错误的 RSA 运算结果，计算出 RSA 私钥。此攻击表明，Rauzy 和 Guilley 的两个 RSA 安全实现算法不能抵抗故障注入攻击。

关键词：RSA 密码；中国剩余定理；侧信道攻击；故障注入攻击

中图分类号：TP309

文献标识码：A

doi: 10.11959/j.issn.2096−109x.2019004

Fault-injection attack on countermeasure algorithms of RSA-CRT cryptosystem

KONG Fanyu[1], QIAO Yong[2], LIU Pengtao[3], LIU Xiaodong[1], ZHOU Dashui[1]

1. Institute of Network Security, Shandong University, Jinan 250100, China
2. China Standard Software Co., Ltd., Beijing 100190, China
3. School of Cyberspace Security, Shandong University of Political Science and Law, Jinan 250014, China

Abstract: As a widely-applied public-key cryptosystem in TLS, SSL and IPSec protocols, the security of RSA cryptosystem is of great importance. At FDTC 2014, Rauzy and Guilley proposed several improved countermeasure algorithms of RSA implementation based on Chinese remainder theorem, which were used to defeat fault-injection attacks. New fault-injection attacks on two of their countermeasure algorithms are proposed. During the RSA computation process, a permanent fault is injected and then a faulty RSA signature result is induced. The RSA private key can be obtained by using the faulty RSA signature and the correct result. Therefore, Rauzy and Guilley's two countermeasure algorithms cannot resist our fault-injection attack.

Key words: RSA cryptosystem, Chinese remainder theorem, side channel attack, fault-injection attack

收稿日期：2018−11−28；**修回日期：**2018−12−28

通信作者：孔凡玉，fangyukong@sdu.edu.cn

基金项目：国家自然科学基金项目资助（No.61602275）

Foundation Item: The National Natural Science Foundation of China (No.61602275)

论文引用格式：孔凡玉, 乔咏, 刘蓬涛, 等. RSA-CRT 密码防御算法的故障注入攻击[J]. 网络与信息安全学报, 2019, 5(1): 30-36.
KONG F Y, QIAO Y, LIU P T, et al. Fault-injection attack on countermeasure algorithms of RSA-CRT cryptosystem [J]. Chinese Journal of Network and Information Security, 2019, 5(1): 30-36.

1　引言

自 Diffie 和 Hellman 提出第一个公钥密码体制——Diffie-Hellman 密钥协商协议[1]以来，公钥密码体制在数字签名、密钥协商等方面发挥了越来越重要的作用。RSA 密码算法[2]是最早提出的几个公钥密码体制之一，其安全性基于大整数分解困难问题，可用于公钥加密和数字签名等功能，是几十年以来应用最广泛的公钥密码体制之一。近年来，基于椭圆曲线的 SM2 密码算法标准和美国的 ECDSA 数字签名标准逐渐推广，但在 IPSec 协议、TLS/SSL 协议以及浏览器等互联网环境中，RSA 密码体制的实际应用依然很广泛，其安全性分析非常重要。

长期以来，密码攻击者主要从数学计算的角度分析一个密码算法的安全性，如通过大整数分解等方法攻击 RSA 密码体制。但是，Kocher 等学者提出的侧信道攻击[3-5]拓展了传统密码分析的思路，能够利用密码算法实际执行时泄露的运算时间、电量消耗等信息恢复部分甚至整个秘密密钥。被动式的侧信道攻击通过测量密码运算的执行时间、电量消耗等信息分析密钥，不影响密码运算的正常执行；主动式的侧信道攻击则可以改变密码运算的执行过程，使其产生对密码分析有用的信息，故障注入攻击是一种主动式攻击方法。

Dan Boneh 首次提出了故障注入攻击[4]的概念，并提出了针对 RSA 密码的中国剩余定理实现（RSA-CRT）的攻击方法：当其中一个模幂运算发生错误时，可以通过错误的 RSA 签名结果恢复 RSA 私钥。Shamir[6]通过在 RSA 运算过程中引入一个随机数 r 来判断两次模幂的结果是否存在错误，如果检测到错误，则只返回错误提示，不返回错误的 RSA 签名值，因此避免攻击者使用错误的签名值恢复私钥。Joye 等学者[7]进一步改进、细化了 Shamir 的方法，使用 d_p 和 d_q 替代私钥指数 d。Aumüller 等[8]指出，在 Shamir 的方法中，如果计算两次模幂的过程中不出错，而是在中国剩余定理的结果合成运算中注入错误，使其中一个模幂结果出错，则仍然能够成功攻击。Yen 等[9]详细分析了 Shamir 的方法和 Aumüller 的改进方法，对 RSA-CRT 的故障注入攻击方法进行了系统的论述。

在 FDTC 2014 会议上，Rauzy 和 Guilley[10]对基于检测和基于感染计算的两类故障注入防御算法进行了分析，并提出了多个抵抗故障注入攻击的算法。2016 年，Battistello 和 Giraud[11]对 Rauzy 和 Guilley 提出的感染计算的防御算法进行了分析，证明其不能抵抗故障注入攻击。关于 RSA-CRT 密码算法的故障注入攻击和防御措施，可以参见相关参考文献[12-22]。

本文针对 Rauzy 和 Guilley 提出的两个基于检测的 RSA-CRT 安全防御算法，进行故障注入攻击，在 RSA 密码运算过程中注入一个永久性错误，使防御算法不能检测到错误的发生，然后利用错误的 RSA 签名运算结果，计算出 RSA 私钥。此攻击表明，Rauzy 和 Guilley 的 RSA 安全实现算法不能抵抗故障注入攻击。

2　RSA-CRT 密码算法和故障注入攻击

本节介绍 RSA 密码算法的中国剩余定理实现，以及针对 RSA-CRT 实现的故障注入攻击和防御措施。

2.1　RSA 密码和中国剩余定理实现

RSA 密码算法[2]是基于整数分解困难问题的公钥密码体制，其公钥包括模数 N 和公钥指数 e，其中 $N=pq$，为了保证安全性，要求 p 和 q 为 512 bit 以上长度的素数。RSA 密码的私钥为 p、q 和私钥指数 d，满足 $e\cdot d\equiv 1 \bmod \varphi(N)$，其中，$\varphi(N)=(p-1)\cdot$

(q–1)是欧拉函数。

RSA 加密或验证签名的过程，即计算模幂 $M^e \bmod N$；RSA 解密或签名的过程，即计算模幂 $M^d \bmod N$。为了提高加密和验签的速度，一般选择比较短的公钥指数 e，如 e=65 537，私钥指数 d 是与模 N 长度相同或接近的整数，RSA 私钥运算 $S=M^d \bmod N$ 的时间复杂性为 $O(\log^3(N))$。

采用中国剩余定理可以提高 RSA 私钥运算的速度（简称 RSA-CRT），即先计算两个模幂 $S_p=M^{dp} \bmod p$ 和 $S_q=M^{dq} \bmod q$，然后组合成最后的计算结果。

$$S=\mathrm{CRT}(S_p, S_q)=S_q+((S_p-S_q)(q^{-1} \bmod p) \bmod p)\cdot q$$

其中，$d_p=d \bmod (p-1)$，$d_q=d \bmod (q-1)$。由于素数 p 和 q 的长度为模 N 的一半，因此每次 S_p 或 S_q 的计算量约为计算 $S=M^d \bmod N$ 的 $\frac{1}{8}$，所以采用中国剩余定理能够提高约 4 倍速度。

2.2 RSA-CRT 密码的故障注入攻击

当 RSA-CRT 运算中的一个模幂运算 $S_p=M^{dp} \bmod p$ 发生错误，即得到错误的模幂结果 S'_p，那么最后的 RSA-CRT 运算结果为

$$S'=\mathrm{CRT}(S'_p, S_q)=S_q+((S'_p-S_q)(q^{-1} \bmod p) \bmod p)\cdot q$$

根据 RSA-CRT 的正确运算结果 S 和错误结果 S'，可以通过欧几里得算法计算出素数 q，即计算[4]

$$q=\mathrm{GCD}((S-S') \bmod N, N)$$

或者由 RSA-CRT 的错误结果 S'，直接计算出素数 q，即计算

$$q=\mathrm{GCD}((S'^{\mathrm{e}}-M) \bmod N, N)$$

存在两大类抵抗故障注入攻击的 RSA-CRT 实现算法：一类是 Shamir[6]提出的基于错误检测的方法，随机选择一个随机数 r，用于盲化素数 p 和 q，先计算模 pr 和 qr 的模幂，然后通过模 r 运算判断两次模幂的结果是否存在错误，如果判断没有错误，则正常计算并返回结果 RSA 签名 S，否则，返回计算失败；另一类是基于感染计算的方法[12]，即不管是否存在模幂运算错误，都返回 RSA 签名结果，但利用错误的 RSA 签名结果不能计算出秘密素数 p 或 q。文献[10,17]对 RSA-CRT 密码的故障注入攻击和防御方法进行了详细论述。

3 Rauzy 等提出的 RSA-CRT 密码防御算法

Rauzy 和 Guilley 在 FDTC 2014 会议上的论文[10]系统地分析了基于检测和基于感染计算的两类故障注入防御算法，指出这两类算法之间可以通过一定的方法相互转化。Rauzy 和 Guilley[10]认为，基于感染计算的抵御算法比基于检测的算法更好，因为它避免了分支判断，减少了故障注入的可能性，同时提出了多个抵抗故障注入攻击的算法。但 Battistello 和 Giraud[11]对 Rauzy 和 Guilley 提出的两个感染计算的防御算法进行了分析，表明其不能抵抗故障注入攻击。

本文主要对 Rauzy 和 Guilley 提出的基于检测的防御算法[10]进行分析，一个被称为“一种直接的防御算法”（原文中的 Algorithm 9），另一个是改进的 Shamir 方法（原文中的 Algorithm 10）。下面分别对这两个算法进行简要描述。

3.1 一种直接的 RSA-CRT 防御算法

Rauzy 和 Guilley 提出的“一种直接的防御算法”（参考文献[10]中的 Algorithm 9）是对模幂 S_p、S_q 和 CRT 绑定结果均进行验证，并被证明能够抵抗一阶故障注入攻击，其算法描述如下。

算法 1 一种直接的 RSA-CRT 防御算法[10]

输入：消息 M，密钥（p, q, d_p, d_q, i_q）

输出：RSA 签名值 $M^d \bmod N$，或返回失败

Begin

Step 1　$S_p=M^{dp \bmod \varphi(p)} \bmod p$；

Step 2　if　$S_p \neq M^{dp} \bmod p$ then return error；

Step 3　$S_q = M^{dq \bmod \varphi(q)} \bmod q$；

Step 4　if $S_q \neq M^{dq} \bmod q$ then return error；

Step 5　$S=\mathrm{CRT}(S_p, S_q)=S_q+q\cdot(i_q\cdot(S_p-S_q) \bmod p)$；

Step 6　if　$((S \neq S_p \bmod p)$ or $(S \neq S_q \bmod q))$ then return error；

Step 7　return S；

End

3.2　改进的 Shamir 防御算法

Rauzy 和 Guilley 提出的改进的 Shamir 方法（参考文献[10]中的 Algorithm 10）对原来的 Shamir 方法进行了修改，并被认为能够抵抗故障注入攻击，其算法描述如下。

算法 2　RSA-CRT 实现的改进 Shamir 防御算法[10]

输入：消息 M，密钥（p, q, d_p, d_q, i_q）

输出：RSA 签名值 $M^d \bmod N$，或返回失败

Begin

Step 1　选择一个小的随机数 r；

Step 2　$p'=p\cdot r$；

Step 3　$q'=q\cdot r$；

Step 4　if $((p'\neq 0 \bmod p)$ or $(q'\neq 0 \bmod q))$ then return error；

Step 5　$S'_p=M^{d \bmod \varphi(p_r)} \bmod p'$；

Step 6　$S'_q=M^{d \bmod \varphi(q_r)} \bmod q'$；

Step 7　if $S'_p \neq S'_q \bmod r$ then return error；

Step 8　$S_p= S'_p \bmod p$；

Step 9　$S_q= S'_q \bmod q$；

Step 10　$S = \mathrm{CRT}(S_p, S_q) = S_q+ q\cdot(i_q\cdot(S_p - S_q) \bmod p)$；

Step 11　if $((S\neq S'_p \bmod p)$ or $(S\neq S'_q \bmod q))$ then return error；

Step 12　return S；

End

4　针对 Rauzy 等的 RSA-CRT 防御算法的攻击

Yen 等[9]将故障注入攻击中发生的错误分为两类：一类是暂时性的错误，即仅在这一次运算时某个数据临时出错，之后恢复原来的正确值；另一类是永久性的错误，即在某个时间点修改了某个数据后，这个错误值将一直保持到整个 RSA 运算过程结束。对 RSA-CRT 密码算法的故障注入攻击方法，包括在 RSA 运算过程中的哪个时间点注入错误，以及注入的是临时性错误还是永久性错误。根据注入错误的次数，可以分为一阶故障注入攻击、二阶故障注入攻击等，分别表示攻击者能够在 RSA 运算过程中注入一个、两个以及更多的错误。

本文主要对 Rauzy 和 Guilley 提出的基于检测的防御算法的故障注入攻击，采用了一阶故障注入攻击方法，产生的是一个永久性错误，并能够顺利通过防御算法的错误检测流程，从而导致产生、输出一个错误的 RSA 签名结果，以用于求解 RSA 私钥。下面分别对这两个算法的攻击方法进行描述。

4.1　对算法 1 的故障注入攻击

Rauzy 和 Guilley 提出的“一种直接的防御算法”（算法 1），分别在 Step 2 和 Step 4 对模幂 S_p、S_q 结果进行错误检测，在 Step 5 进行 CRT 组合运算，然后在 Step 6 检测 CRT 组合运算结果是否有错误。

本文提出的故障注入攻击方法是在 Step 5 计算过程中，对 S_p 注入一个永久性错误，攻击方法描述如下。

攻击方法 1　针对算法 1 的故障注入攻击

输入：消息 M，密钥（p, q, d_p, d_q, i_q）

输出：RSA 签名值 $M^d \bmod N$，或返回失败

Begin

Step 1　$S_p=M^{dp \bmod \varphi(p)} \bmod p$；

Step 2 if $S_p \neq M^{dp} \bmod p$ then return error；

Step 3 $S_q = M^{dq \bmod \varphi(q)} \bmod q$；

Step 4 if $S_q \neq M^{dq} \bmod q$ then return error；

Step 5 对 S_p 注入一个永久性错误，即将其修改为 S'_p，计算：

$S' = \mathrm{CRT}(S'_p, S_q) = S_q + q \cdot (i_q \cdot (S'_p - S_q) \bmod p)$；

Step 6 if $((S' \neq S'_p \bmod p)$ or $(S' \neq S_q \bmod q))$ then return error；

Step 7 return S'；

End

下面说明故障注入攻击方法 1 能够成功恢复 RSA 私钥。

首先，本文的攻击方法不改变 Step1~Step4 的运算，因此 Step1~Step4 能够顺利执行，不会返回错误。然后，在 Step 5 中，对 S_p 注入一个永久性错误，即将其修改为 S'_p，并计算 $S' = \mathrm{CRT}(S'_p, S_q)$。因为 S'_p 是一个永久性错误，CRT 组合运算只是按照上面的运算公式计算 S'，其结果 S'依然满足 $S' = S'_p \bmod p$ 和 $S' = S_q \bmod q$，因此 Step 6 并不能检测到 S'_p 的错误，Step 7 会返回一个错误的 RSA 签名值 S'。根据错误的 RSA 签名值 S'，容易计算出素数 $q = \mathrm{GCD}((S'^e - M) \bmod N, N)$，从而恢复 RSA 私钥。

4.2 对算法 2 的故障注入攻击

Rauzy 和 Guilley 提出的改进的 Shamir 方法（算法 2），在 Step 4 检测 p'和 q'的正确性，在 Step 7 检测模幂 S'_p、S'_q 的正确性，在 Step 10 进行 CRT 组合运算，然后在 Step 11 检测 CRT 组合运算结果是否存在错误。

本文提出的故障注入攻击方法是在 Step 8 计算过程中，对 S'_p 注入一个永久性错误，攻击方法描述如下。

攻击方法 2 针对算法 2 的故障注入攻击

输入：消息 M，密钥（p, q, d_p, d_q, i_q）

输出：RSA 签名值 $M^d \bmod N$，或返回失败

Begin

Step 1 选择一个小的随机数 r；

Step 2 $p' = p \cdot r$；

Step 3 $q' = q \cdot r$；

Step 4 if $((p' \neq 0 \bmod p)$ or $(q' \neq 0 \bmod q))$ then return error；

Step 5 $S'_p = M^{d \bmod \varphi(p')} \bmod p'$；

Step 6 $S'_q = M^{d \bmod \varphi(q')} \bmod q'$；

Step 7 if $S'_p \neq S'_q \bmod r$ then return error；

Step 8 对 S'_p 注入一个永久性错误，即将其修改为 S'^*_p，计算：$S^*_p = S'^*_p \bmod p$；

Step 9 $S_q = S'_q \bmod q$；

Step 10 $S^* = \mathrm{CRT}(S^*_p, S_q) = S_q + q \cdot (i_q \cdot (S^*_p - S_q) \bmod p)$；

Step 11 if $((S^* \neq S'^*_p \bmod p)$ or $(S^* \neq S'_q \bmod q))$ then return error；

Step 12 return S^*；

End

下面说明故障注入攻击方法 2 能够成功恢复 RSA 私钥。

首先，本文的攻击方法不改变 Step1~Step7 的运算过程，因此不会返回错误。然后，在 Step 8 中，对 S'_p 注入一个永久性错误，即将其修改为 S'^*_p，并计算出一个错误的模幂值 $S^*_p = S'^*_p \bmod p$。在 Step 10 中，CRT 组合运算将错误的模幂 S^*_p 和正确的模幂 S_q 进行组合计算，得到错误的 S^*。因为在 Step 8 中，S^*_p 是由 S'^*_p 模 p 计算出来的，因此 Step 11 中的 S^*满足 $S^* = S'^*_p \bmod p$ 和 $S^* = S'_q \bmod q$。所以，Step 12 会返回一个错误的 RSA 签名值 S^*。根据错误的 RSA 签名值 S^*，容易计算出素数 $q = \mathrm{GCD}((S^{*e} - M) \bmod N, N)$，从而恢复 RSA 私钥。

不管在硬件还是软件实现中，S_p 和 S'_p 等大整数都是以数组的形式存储在内存或寄存器中，如在 Intel CPU 中，可以是以 32 或 64 比特为单

位的数组进行存储。所以，本文提出的故障注入攻击，只要对 S_p 和 S'_p 等数据的某一个 32 或 64 比特存储单位进行修改，就可以实现相应的故障注入攻击。

5　结束语

故障注入攻击等侧信道攻击方法表明，一个密码算法的正确、安全实现与其安全性密切相关。针对 RSA-CRT 密码实现的故障注入攻击是一种很强大的攻击技术，仅需要一次模幂出现错误即可攻击成功，因此必须采用安全的实现算法以抵抗故障注入攻击。

本文对 Rauzy 和 Guilley 提出的两个故障注入攻击算法进行了分析，提出了一阶故障注入攻击算法，表明这两个算法不能抵抗故障注入攻击。如何在抵抗故障注入攻击的同时，提高 RSA-CRT 密码实现的速度，并分析在手机、物联网、车联网等移动环境下的实际攻击技术，是下一步需要研究的方向。

参考文献：

[1] DIFFIE W, HELLMAN M E. New directions in cryptography[J]. IEEE Trans Inform Theory, 1976, 22(6): 644-654.

[2] RIVEST R L, SHAMIR A, ADLEMAN M. A method for obtaining digital signatures and public-key cryptosystems[J]. Communications of the ACM, 1978, 21(2): 120-126.

[3] KOCHER P. Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems[C]//Advances in Cryptology-CRYPTO'96. 1996: 104-113.

[4] BONEH D, DEMILLO R A, LIPTON R J. On the importance of checking cryptographic protocols for faults[C]//Advances in Cryptology - EUROCRYPT 1997. 1997: 37-51.

[5] KOCHER P, JAFFE J, JUN B. Differential power analysis[C]//Advances in Cryptology - CRYPTO'99. 1999: 388-397.

[6] SHAMIR A. Method and apparatus for protecting public key schemes from timing and fault attacks [P].1999.

[7] JOYE M, PAILLER P, YEN S M. Secure evaluation of modular functions[C]//2001 International Workshop on Cryptology and Network Security. 2001: 227-229.

[8] AUMÜLLER C, BIER P, FISCHER W, et al. Fault attacks on RSA with CRT: concrete results and practical countermeasures [C]//Cryptographic Hardware and Embedded Systems-CHES 2002. 2002: 260-275.

[9] YEN S M, MOON S J, HA J. Hardware fault attack on RSA with CRT revisited[C]//Information Security and Cryptology-ICISC 2002. 2002: 374-388.

[10] RAUZY P, GUILLEY S. Countermeasures against high-order fault-injection attacks on CRT-RSA [C]//Fault Diagnosis and Tolerance in Cryptography 2014. 2014: 68-82.

[11] BATTISTELLO A, GIRAUD C. Lost in translation: fault analysis of infective security proofs[C]//The Workshop on Fault Diagnosis & Tolerance in Cryptography. 2016: 45-53.

[12] YEN S M, KIM S, LIM S, et al. Moon. RSA speedup with Chinese remainder theorem immune against hardware fault cryptanalysis[J]. IEEE Transactions on Computers, 2003, 52(4): 461-472.

[13] YEN S M, KIM D, MOON S. Cryptanalysis of two protocols for RSA with CRT based on fault infection[C]//Fault Diagnosis and Tolerance in Cryptography 2006. 2006: 53-61.

[14] WAGNER D. Cryptanalysis of a provably secure CRT-RSA algorithm[C]//ACM Conference on Computer and Communications Security 2004. 2004: 92-97.

[15] GIRAUD C. An RSA implementation resistant to fault attacks and to simple power analysis[J]. IEEE Transactions on Computers, 2006, 55(9): 1116-1120.

[16] VIGILANT D. RSA with CRT: a new cost-effective solution to thwart fault attacks[C]//Cryptographic Hardware and Embedded Systems - CHES 2008. 2008: 130-145.

[17] JOYE M, TUNSTALL M. Fault analysis in cryptography[M]. Berlin Heidelberg: Springer, 2012.

[18] LEE S, CHOI D, CHOI Y. Improved Shamir's CRT-RSA algorithm: revisit with the modulus chaining method [J]. ETRI Journal, 2014, 36.3: 469-478.

[19] 王红胜, 纪道刚, 张阳, 等. 针对 RSA-CRT 数字签名的光故障攻击研究[J]. 电子设计工程, 2015, 23(6): 12-5.
WANG H S, JI D G, ZHANG Y, et al. Optical fault attack on RSA-CRT signatures[J]. Electronic Design Engineering, 2015, 23(6): 12-15.

[20] 李增局. 一种关于 CRT-RSA 算法的差分错误注入攻击[J]. 密码学报, 2016, 3(6): 546-554.
LI Z J. Differential fault attack on CRT-RSA [J]. Journal of Cryptologic Research, 2016, 3(6): 546–554.

[21] KISS A, KRMER J, RAUZY P, et al. Algorithmic countermeasures against fault attacks and power analysis for RSA-CRT [C]//Interna-

tional Workshop on Constructive Side-Channel Analysis and Secure Design. 2016: 111-129.

[22] KONG F, ZHOU D S, JIANG Y L, et al. Fault attack on an improved CRT-RSA algorithm with the modulus chaining method [C]//International Conference on Computational Science and Engineering (CSE) 2017. 2017: 866-869.

[作者简介]

孔凡玉（1978− ），男，山东莱芜人，博士，山东大学副教授，主要研究方向为密码学与信息安全。

乔咏（1977− ），男，北京人，中标软件有限公司高级工程师，主要研究方向为系统与软件安全。

刘蓬涛（1980− ），女，山东蓬莱人，山东政法学院副教授，主要研究方向为密码学。

刘晓东（1975− ），男，山东寿光人，博士，山东大学讲师，主要研究方向为密码学。

周大水（1963− ），男，山东潍坊人，山东大学教授，主要研究方向为网络系统安全。

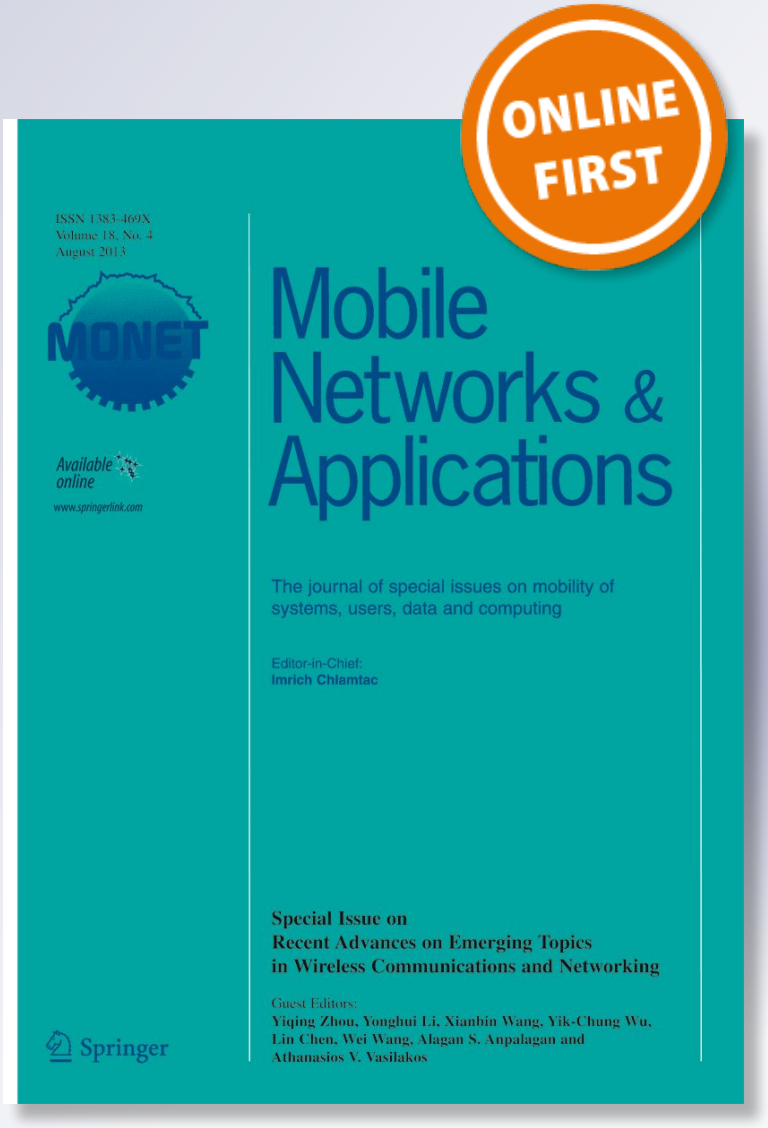
Verifiable Public-Key Encryption with Keyword Search Secure against Continual Memory Attacks

Chengyu Hu, Zhen Li, Pengtao Liu, Rupeng Yang, Shanqing Guo & Hailong Zhang

Mobile Networks and Applications
The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing

ISSN 1383-469X

Mobile Netw Appl
DOI 10.1007/s11036-018-1101-4



Springer

Mobile Networks and Applications
https://doi.org/10.1007/s11036-018-1101-4

Verifiable Public-Key Encryption with Keyword Search Secure against Continual Memory Attacks

Chengyu Hu[1,2] · Zhen Li[3] · Pengtao Liu[4] · Rupeng Yang[5] · Shanqing Guo[1] · Hailong Zhang[6]

Abstract
Public-key encryption with keyword search (PEKS) enables users to search on encrypted data which is applicable to scenario of sharing data in the cloud storage. The existing PEKS schemes fail to verify the returned result from the tester, i.e. they cannot guarantee the correctness and completeness of the result. In this paper, we resolve this problem by constructing a verifiable PEKS scheme which can efficiently verify the completeness of the result and thus the correctness. We also consider the security of verifiable PEKS scheme against a kind of side-channel attacks called the continual memory attacks which allow the adversary to obtain some leakage information of the secret key used in the search trapdoor generation algorithm and can help it break the security of the scheme. We extend our scheme to an enhanced one which remains secure even when the adversary can obtain unbounded total leakage information during the whole lifetime. Moreover, to make it resist keyword guessing attacks, we extend our scheme to one with a designated tester.

Keywords Leakage resilience · Verifiable · PEKS · Continual memory attacks · Keyword guessing attacks

1 Introduction

As more and more IT applications have been shifted to the cloud [21, 43], the amount of sensitive data to be outsourced in the cloud storage has been rapidly increasing. To ensure the privacy and confidentiality of sensitive data from even inside attackers such as a malicious cloud server, a user may encrypt the sensitive data before uploading the data into the cloud and use some mechanisms to securely implement data in ciphertext type [14, 15, 23, 24, 48]. Meanwhile, the cloud server should provide rigorous security mechanisms to assure the users that their data have been securely protected, such as authentication [7], access control [29, 39, 49], auditing [38, 40, 46] and some data protection schemes [3, 4].By these ways, confidentiality of the data is guaranteed. However, storing encrypted data in the cloud makes it difficult to retrieve the data since the encrypted data are unreadable for the cloud server. In this scenario, how to effectively obtain parts of the encrypted data without decrypting them becomes a new security issue. To cope with this problem, the notion of "searchable encryption" is proposed and studied in two versions: symmetric searchable

✉ Chengyu Hu
hcy@sdu.edu.cn

Zhen Li
sdufelizhen@126.com

Pengtao Liu
ptwave@163.com

Rupeng Yang
orbbyrp@gmail.com

Shanqing Guo
guoshanqing@sdu.edu.cn

Hailong Zhang
zhanghailong@iie.ac.cn

[1] School of Software Engineering, Shandong University, Jinan, 250101, China

[2] School of Computer Science and Technology, Anhui University, Hefei, 230601, China

[3] School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan, China

[4] School of Information, Shandong University of Political Science and Law, 250014, Jinan, China

[5] School of Computer Science and Technology, Shandong University, 250101, Jinan, China

[6] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, 100093, Beijing, China

Published online: 04 August 2018

encryption(SSE) [6, 25, 26, 33–37, 44] and public-key encryption with keyword search (PEKS) which was first introduced by [11]. However, although PEKS has received a lot of attention [10, 13, 16, 32, 45, 50] in the past few years, most of them failed to efficiently verify the correctness and completeness of the search result from the cloud server. In addition, these schemes mainly provide security in the leak-free scenario assuming that the secret key is completely hidden from the adversary. However, in a real life, side-channel attacks [8, 9, 22, 27, 30, 31] can recover a fraction of the secret key used in trapdoor generation to the adversary and break the security of a PEKS scheme. To the best of our knowledge, no existing PEKS scheme is adequate for achieving both of the "verifiability" and "leakage-resilience" goals simultaneously.

1.1 Related works

Boneh et al. [11] introduced the concept of "Public-Key Encryption with Keyword Search" and proposed a universal transformation to construct PEKS from identity-based encryption (IBE) [42]. Abdalla et al. [1] stated the formal definition of anonymity of IBE and showed that it is the anonymity of IBE that ensures a PEKS ciphertext reveal no information about the keyword. After that, there are various PEKS schemes proposed to search on complex conditions. For examples, Boneh and Waters [10] proposed PEKS scheme which supports conjunctive, subset, and range comparisons over the keywords. Lai et al. [32] presented a more expressive and efficient construction. Zhu et al. [50] considered the predicate privacy in PEKS and the works in [13, 16, 41] studied the off-line keyword guessing attacks on PEKS. As the search is intended to be applied to outsourced encrypted data in the cloud storage, it is unacceptable for these schemes to fail to verify whether the cloud server has faithfully implemented the requested search operations and returned all of the results. Zheng et al. [47] firstly addressed this problem in the public-key setting and proposed a novel cryptographic primitive, called verifiable attribute-based keyword search (VABKS) to resolve it. However, its verification algorithm needs both the secret key and the trapdoor used in current search besides the searched keyword, and implements some expensive operations which are the same as the cloud server does. In addition, *in our opinion*, the VABKS scheme in this work is not suitable for the data sharing scenario of PEKS as the secret key used to generate search trapdoor is generated by data owner or a trusted third party rather than the data user itself.

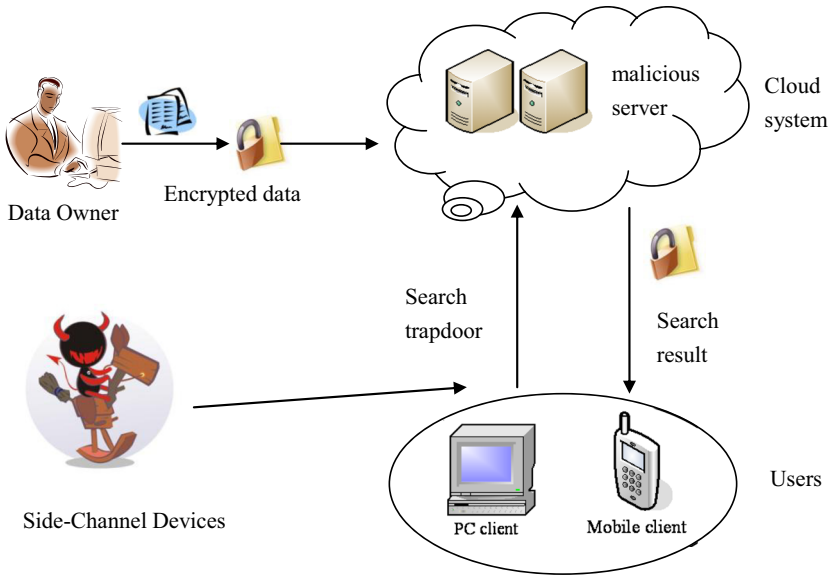
Recently, it is shown that cryptosystems proved secure in the traditional model may not resist the side-channel attacks [8, 9, 22, 27, 30, 31] which can recover a fraction of the secret key to the adversary. For a PEKS scheme, side-channel attacks can recover a fraction of the secret key used in *trapdoor generation* to the adversary and break its security. To protect against above realistic attacks in the provable security approach, one can model them using abstract notions of computation which leads to many studies made in leakage-resilient cryptography. Dziembowski and Pietrzak [20] presented the first leakage-resilient cryptographic scheme assuming that "*only computation leaks information*" which fails to capture the "cold boot" attack [27] and other attacks in which information leaks from all parts of memory. To tackle this type of memory leakage attacks, Akavia et al. [2] proposed the "*bounded memory leakage model*" by considering adversaries that can obtain leakage from all parts of the memory even if they are not currently used in any computation. However, they bound the overall amount of leakage throughout the lifetime of the scheme which seems unrealistic since attackers in the real world may employ multiple attacks over time. Therefore, Brakerski et al. [12] proposed the "*continual memory leakage model*" (CML) which partitions the entire lifetime of the scheme into discrete time periods, and allows the adversary to obtain bounded leakage from the entire internal secret state during each time period while the total leakage over the lifetime of the scheme is unbounded. Focusing on constructing leakage-resilient PEKS schemes, Chen et al. [17] introduced the notion of leakage-resilient anonymity of PEKS scheme in the bounded memory leakage model, and constructed leakage-resilient anonymous PEKS scheme through their leakage-resilient anonymous IBE scheme. Hu et al. [28] considered the continual memory leakage resilience of PEKS in the trapdoor generation algorithm and provide a method of constructing continual leakage-resilient PEKS schemes. However, their scheme fails to verify the completeness of returned results and allows no leakage in the key-update algorithm.

1.2 Our contribution

In this paper, we focus on how to construct a *verifiable* public-key encryption with keyword search scheme (VPEKS) secure against *continual memory attacks* which allows information leakage in the key-update algorithm besides the secret key leakage in the trapdoor generation algorithm. We show in Fig. 1 the attack scenario considered in this work and summarize our contributions in the following:

– Propose a method of transforming an anonymous IBE scheme to a verifiable PEKS scheme allowing the verification of the completeness of returned result, which we call *IBE-to-VPEKS*. Specifically, we first apply IBE-to-PEKS transformation in [1] to an anonymous IBE scheme. In this transformation, the cipher of keyword w includes (C, R) where $C \leftarrow$IBE.Enc(pk, w, R) and

Fig. 1 Attack scenario in Cloud



R is a random message. To enable it to verify the search result, we add to the keyword cipher a value computed by a pseudorandom function F taking the hash value of all the data ciphers associated with the search keyword as the key and taking a randomness extracted from w, R as its input, i.e. $F(Hash(allcipher), randExt(w,R))$ where randExt is a *randomness extractor* and $Hash(allcipher)$ is the *hash value of all the data ciphers* associated with the search keyword w. The cloud server should return data ciphers with their respective R and $f=F(Hash(allcipher), randExt(w,R))$. Then, the data user can compute $f'=F(Hash(allcipher), randExt(w,R))$ for each result itself and verify the completeness of the results by checking the equality of each f and f'.

- Extend our VPEKS scheme to resist continual memory attacks *with leakage on key updates*. To this end, we first show that if the based anonymous IBE scheme is continual master-key leakage-resilient secure, then the VPEKS scheme constructed from it via *IBE-to-VPEKS* transformation is continual leakage-resilient secure. Then, we extend above VPEKS scheme to one that allows leakage on key updates using the techniques introduced in [18]. Specifically, we extend the notion of "consecutive" CLR or 2CLR to IBE and VPEKS, and prove that the concrete anonymous IBE scheme presented in [28] is 2CLR secure. By compiling the the key update algorithm of anonymous IBE scheme presented in [28] using the techniques introduced in [18], a continual leakage-resilient IBE with leakage on key updates can be obtained. At last, VPEKS scheme obtained as above is continual leakage secure with leakage on key updates.
- Extend our VPEKS scheme to a continual leakage-resilient verifiable designated tester public-key encryption with keyword search (dPEKS) secure against *keyword guessing attacks*. We extend the dPEKS definition to verifiable dPEKS (VdPEKS) and propose a transformation called *(IBE,TE)-to-VdPEKS* which can transform an IBE scheme and a traditional PKE scheme (TE) to a verifiable dPEKS scheme. We prove that if the underlying IBE scheme is continual leakage-resilient secure, then the obtained VdPEKS scheme is continual leakage-resilient secure.

1.3 Organization

In Section 2, we review some preliminaries and describe the continual leakage-resilient security model for anonymous IBE. In Section 3, we give the definition of verifiable public-key encryption with keyword search (VPEKS) secure against continual memory attacks, and show how to construct it. In Section 4, we extend the VPEKS scheme to one that is secure against keyword guessing attacks. Finally, we draw our conclusions in Section 5.

2 Preliminaries

In this section, we recall some basic notions, terminology and computational assumption. Also, we define the anonymous identity-based encryption with key-update and describe the leakage-resilient security model of it.

2.1 Randomness extractor and pseudorandom function

We recall some basic notions relating to randomness extractors here. Let X and Y be two random variables in a finite set U. The *statistical distance* between X and Y is

defined as SD(X,Y)=$\frac{1}{2}\sum_{u\in U}|Pr[X=u]-Pr[Y=u]|$. We say that two variables are $\epsilon-close$ if their statistical distance is at most ϵ. Let X be a random variable. Then the min-entropy of X, denoted as $H_\infty(X)$ is defined as $H_\infty(X)=-log(max_x Pr[X=x])$. Let X and Y be two random variables. Then the average min-entropy of X conditioned on Y, denoted as $\widetilde{H}_\infty(X|Y)$, is defined as

$$\begin{aligned}\widetilde{H}_\infty(X|Y) &= -log(\mathbb{E}_{y\leftarrow Y}[max_{x\leftarrow X}Pr[X=x|Y=y]])\\ &= -log(\mathbb{E}_{y\leftarrow Y}[2^{-H_\infty(X|Y=y)}]).\end{aligned}$$

Definition 1 (Randomness Extractor) [19] Let U be a finite set. A function Ext:$U\times\{0,1\}^t\rightarrow\{0,1\}^m$ is an average-case (v,ϵ)-strong extractor if for all pairs of random variables (X, I) such that the range of X is U and $\widetilde{H}_\infty(X|I)\geq v$, it holds that:

$$\mathrm{SD}((\mathrm{Ext}(X,R),R,I),(U_m,R,I))\leq\epsilon$$

where R is uniform on $\{0,1\}^t$.

For the existence of average-case randomness extractors, Dodis et al. [19] proved that from any family of universal hash functions we can get an average-case strong extractor. More precisely, we have:

Lemma 1 *Fix an output length m, for any* $v\geq 0$ *and* $\epsilon\geq 0$*, we have an average-case (v,*ϵ*)-strong extractor from* $U\times\{0,1\}^t$ *to* $\{0,1\}^m$ *as long as* $m\leq v-2log(\frac{1}{\epsilon})+2$.

Definition 2 (Pseudorandom Function) Let $F:\{0,1\}^*\times\{0,1\}^*\rightarrow\{0,1\}^*$ be an efficient, length-preserving, keyed function. We say F is a pseudorandom function if for all probabilistic polynomial-time distinguishers D, there exists a negligible function $negl$ such that:

$$|Pr[D^{F_k(\cdot)}(1^n)=1-|Pr[D^{f_n(\cdot)}(1^n)=1|\leq \mathrm{negl}(n),$$

where $k\leftarrow\{0,1\}^n$ is chosen uniformly at random and f_n is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.

2.2 Anonymous identity-based encryption with key-update

We describe the definition of anonymous identity-based encryption with key-update [28] in this section. An identity-based encryption scheme with key-update consists of five algorithms as follows:

- Setup(λ): takes as input a security parameter λ, it outputs the public key pk and a master key msk.
- KeyGen(msk,ID): takes as input the master key msk and an identity ID. It outputs a secret key sk_{ID}.
- Encrypt(pk,ID,m): takes as input the public key pk, an identity ID, and a message m. It outputs a ciphertext CT.
- Decrypt(pk,sk_{ID},CT): takes as input public key pk, the private key sk_{ID} and a ciphertext CT=Encrypt (pk,ID,m). It outputs either a message m except with negligible probability or a reject symbol $\perp$ indicating CT is invalid.
- UpdateKey(sk,ID): takes as input the master key(ID =ϵ) or a secret key of an identity ID. It outputs a re-randomized key sk', such that $|sk|=|sk'|$ and the distribution of sk is indistinguishable from the distribution of sk'.

The correctness of IBE requires that for any ID and any m, we have:

$$\begin{aligned}\Pr[\mathrm{Decrypt}(\mathrm{KeyGen}(msk,ID)/\mathrm{UpdateKey}(sk,ID),\\ \mathrm{Encrypt}(pk,ID,m))\neq m]\leq \mathrm{negl}(\lambda)\end{aligned}$$

where the probability is taken over random coins used by Setup, KeyGen and Encrypt.

Security model for anonymous IBE The security definition of the anonymous IBE which captures semantic security and anonymity by means of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- Setup: The challenger $\mathcal{C}$ takes a security parameter 1^λ and runs the Setup algorithm. The public key pk and the system parameters are given to $\mathcal{A}$. The challenger $\mathcal{C}$ keeps the master secret key msk to itself. The challenger will initialize a set $S=\emptyset$, which will be the set of private keys it has created, but not given out. The challenger will also initialize a set $R=\emptyset$, which will be the set of identities and private keys it has revealed.
- Phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries:

  Create Queries. The adversary gives an identity ID to the challenger. The challenger creates a key for the identity, but does not give it to the adversary. It instead adds the key to the set S and gives the adversary a reference to it.

  Reveal Queries. The adversary specifies an element of the set S for a private key sk. The challenger removes the item from the set S and adds the identity and the private key into set R. Then the challenger gives the adversary the private key.

- Challenge: $\mathcal{A}$ outputs two pairs of message and identity (M^*, ID_0^*) and (M^*, ID_1^*) to $\mathcal{C}$ where $ID_0^*, ID_1^*\notin R$. $\mathcal{C}$ chooses a random bit $\beta\in\{0,1\}$, encrypts M^* under ID_β^* and sends the resulting ciphertext to $\mathcal{A}$.
- Phase 2: This is the same as Phase 1 with the added restriction that only queries allowed are *Create* and *Reveal* queries that involve secret key with identity different than ID_0^*, ID_1^*.

– Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$. It wins the game if $\beta' = \beta$.

The advantage of the adversary $\mathcal{A}$ in this game is defined as $|\Pr[\beta = \beta']\text{-}1/2|$.

Definition 3 An IBE scheme is anonymous and semantically secure against chosen plaintext attacks (ANO-IND-ID-CPA) if all polynomial-time adaptive adversaries $\mathcal{A}$ have at most negligible advantage in the above game, where $\mathcal{A}$'s advantage is defined as

$$\mathrm{Adv}_{IBE,\mathcal{A}}^{ANO-IND-ID-CPA}(1^\lambda) = |\Pr[\beta = \beta'] - 1/2|.$$

Leakage resilience To satisfy the continual leakage-resilient security of anonymous IBE, *Leak* queries are added which allow leakage on the master key and secret keys in Phase 1 of the above game while forbidding the *Leak* query in Phase 2. The only restriction is that the adversary can not get leakage of more than l_{MSK} bits of the master key and l_{SK} bits of each secret key where l_{MSK}, l_{SK} are parameters of the game. The security of continual leakage-resilient anonymous IBE is based on the following game (**MasterLeak**) between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

– Setup: The challenger $\mathcal{C}$ takes a security parameter 1^λ and runs the Setup algorithm. The public key pk and the system parameters are given to $\mathcal{A}$. The challenger $\mathcal{C}$ keeps the master secret key msk to itself. The challenger will initialize a set $S = \{(0, \epsilon, msk, 0)\}$, which will be the set of tuples of handles, identities, secret keys it has created and the number of leaked bits, i.e. (h, ID, sk_{ID}, L_{sk}) or $(h, \epsilon, msk, L_{MSK})$, but not revealed. Here, the set S initially holds a record of the original master key. The challenger will also initialize a set $R = \emptyset$, which will be the set of identities whose secret keys have been revealed. Let L_{MSK} be the number of bits that have been leaked with the master key and L_{sk} be the number of bits that have been leaked with the secret key sk_{ID}.
– Phase 1: The adversary $\mathcal{A}$ adaptively issues the following three kinds of queries:

  Create Queries. The adversary gives an identity ID to the challenger. The challenger creates a key for the identity, but does not give it to the adversary. It instead sets $h = h + 1$ and adds the $(h, ID, sk_{ID}, 0)$ to the set S and gives the adversary the handle h to it.[1] If $ID = \epsilon$, then the challenger calls msk'=UpdateKey(msk), sets $h = h + 1$ and adds the $(h, \epsilon, msk', 0)$ to the set S and gives the adversary the handle h to it.

  Leak Queries. The adversary gives a polynomial-time computable arbitrary function $f : \{0, 1\}^* \to \{0, 1\}^*$ with a queried handle h of key to the challenger. If the key of the queried handle is master key, the challenger checks if $L_{MSK} + |f(msk)| \leq l_{MSK}$. If this is true, it responds with $f(msk)$ and updates L_{MSK} with $L_{MSK} + |f(msk)|$. If the check fails, it returns $\perp$ to the adversary. If the key of the queried handle is a secret key, the challenger finds the tuple (h, ID, sk, L_{sk}) and checks if $L_{sk} + |f(sk)| \leq l_{SK}$. If this is true, it responds with $f(sk)$ and updates L_{sk} with $L_{sk} + |f(sk)|$. If the check fails, it returns $\perp$ to the adversary.

  Reveal Queries. The adversary specifies an element of the set S for a secret key sk.[2] The challenger scans S to find the requested entry. If the handle refers to a master key tuple, then the challenger returns $\perp$. Otherwise, let the tuple be denoted by (h, ID, sk, L_{sk}), the challenger removes the item from the set S and adds the identity ID into set R. Then the challenger gives the adversary the secret key sk.

– Challenge: $\mathcal{A}$ outputs two pairs of message and identity (M^*, ID_0^*) and (M^*, ID_1^*) to $\mathcal{C}$ where $ID_0^*, ID_1^* \notin R$. $\mathcal{C}$ chooses a random bit $\beta \in \{0, 1\}$, encrypts M^* under ID_β^* and sends the resulting ciphertext to $\mathcal{A}$.
– Phase 2: This is the same as Phase 1 with the added restriction that only queries allowed are *Create* and *Reveal* queries that involve secret key with identity different than ID_0^*, ID_1^*.
– Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$. It wins the game if $\beta' = \beta$.

The advantage of the adversary $\mathcal{A}$ in this game is defined as $|\Pr[\beta = \beta']\text{-}1/2|$.

Definition 4 An IBE scheme is (l_{MSK}, l_{SK})-masterkey-leakage anonymous and secure against chosen plaintext attacks ((l_{MSK}, l_{SK})-ANO-IND-ID-CPA) if all polynomial-time adaptive adversaries $\mathcal{A}$ have at most negligible advantage in the above game, where $\mathcal{A}$'s advantage is defined as

$$\mathrm{Adv}_{IBE,\mathcal{A}}^{(l_{MSK},l_{SK})-ANO-IND-ID-CPA}(1^\lambda) = |\Pr[\beta = \beta'] - 1/2|.$$

[1] Create Queries implies the adversary can issue an update query for an identity and the challenger returns the handle of the new secret key of the identity to the adversary. So, an identity may have several handles in set S with respect to all of his secret keys used in entire lifetime.

[2] It is not allowed for the adversary to request the entire master key.

3 Continual leakage-resilient verifiable public-key encryption with keyword search

Hu et al. [28] defined the model of continual leakage-resilience for PEKS (PEKS-CML). In this section, we will equip PEKS-CML with verification algorithm which allows the data user to verify the result and propose a method to get a verifiable PEKS scheme secure against continual memory attacks (VPEKS-CML).

3.1 Definition and security

The model of verifiable PEKS (VPEKS) is: A data owner outsources its data associated with encrypted keywords to the cloud, a data user generates search trapdoors according to some keywords, and the cloud server implements the search operations over outsourced encrypted keywords after receiving search trapdoors from the user. The data user can verify the correctness and completeness of the results returned by the cloud server. The data user can also update its secret key used to generate search trapdoors.

Formally, a verifiable PEKS scheme secure against continual memory attacks (VPEKS-CML) consists of five algorithms, Setup, PEKS, Trapdoor, Test, Verify and UpdateKey:

–Setup(λ): The system setup algorithm generates the public/secret key pair (pk, sk), taking a security parameter λ as input. Let the keyword space be W.

–PEKS (pk,w): The keyword ciphertext generation algorithm encrypts the keyword w which is associated with one document using the public key pk, and generates a searchable ciphertext CT of w for test.

–Trapdoor(sk,w): The trapdoor generation algorithm outputs the trapdoor T_w of a keywords w using the secret key sk.

–Test(pk,CT,T_w): The test algorithm checks whether a ciphertext CT and a trapdoor T_w associate with the same keyword using the public key pk. If it is true, the algorithm outputs 1. Otherwise, the algorithm outputs 0. Let Rst_w be the results of the search about keyword w returned by the tester.

–Verify(Rst_w,w): The verify algorithm verify the completeness of the result Rst_w. It output 1 if the tester returns all the data associated with keyword w, and output 0 otherwise.

–UpdateKey(sk): The updatekey algorithm takes as input a secret key sk and outputs a re-randomized key sk'. The length of the new key sk' should be equal to the old key sk (i.e. $|sk| = |sk'|$) and the distribution of sk is indistinguishable from the distribution of sk'.

It is required that for any $w, w' \in W$,

$$\Pr[\text{Test}(pk, \text{PEKS}(pk, w), \text{Trapdoor}(sk, w)) = 1] \geq 1 - \text{negl}(\lambda),$$

and

$$\Pr[\text{Test}(pk, \text{PEKS}(pk, w), \text{Trapdoor}(sk, w')) = 1 : w \neq w'] \leq \text{negl}(\lambda).$$

The security for verifiable PEKS scheme secure against continual memory attacks is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

–Setup: Taking as input a security parameter 1^λ, the challenger $\mathcal{C}$ runs the Setup algorithm and generates a key pair (pk,sk) with a keyword space W. The public key pk and the keyword space W are given to $\mathcal{A}$ and the challenger $\mathcal{C}$ retains the secret key sk. Let L_{SK} be the number of bits that have been leaked with the secret key and l_{SK} be the leakage bound.

–Phase 1: The adversary $\mathcal{A}$ adaptively issues the following queries:

Trapdoor Queries. The adversary $\mathcal{A}$ gives a keyword w to the challenger. The challenger generates a trapdoor T_w for the keyword w, then gives it to $\mathcal{A}$.

Leak Queries. The adversary $\mathcal{A}$ gives a polynomial-time computable arbitrary function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ to the challenger. The challenger checks if $L_{SK} + |f(sk)| \leq l_{SK}$. If this is true, it returns $f(sk)$ to $\mathcal{A}$ and updates L_{SK} with $L_{SK} + |f(sk)|$. If the check fails, it returns $\perp$ to $\mathcal{A}$.

Update Queries. The adversary $\mathcal{A}$ requests an update on the secret key. The challenger generates a new secret key, sk'=UpdateKey(sk) and sets sk=sk'.[3]

–Challenge: The adversary $\mathcal{A}$ sends two keywords w_0, w_1 to the challenger $\mathcal{C}$ where $|w_0| = |w_1|$ and they were not previously queried for the *Trapdoor Queries*. $\mathcal{C}$ randomly chooses $b \in \{0,1\}$, and responds to the adversary CT=PEKS (pk,w_b).

–Phase 2: This phase is the same as Phase 1 except that the queried keyword w in *Trapdoor Queries* should not be w_0, w_1 and *Leak Queries* are not allowed.

–Guess: The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

Definition 5 We say that a verifiable PEKS scheme is continual l_{SK}-leakage secure under chosen plaintext attacks (l_{SK}-*cIND-CPA*) if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above game is negligible, where $\mathcal{A}$'s advantage is defined as $\text{Adv}^{l_{SK}-cIND-CPA}_{VPEKS,\mathcal{A}}(1^\lambda) = |\Pr[b' = b]\text{-}1/2|$.

Definition 6 We say that a verifiable PEKS scheme is PEKS cipher indistinguishability secure under chosen plaintext attacks (*IND-CPA*) if for all PPT adversaries $\mathcal{A}$, the

[3] In this definition, we do not allow the leakage in the UpdateKey process.

advantage of $\mathcal{A}$ in the above game is negligible, where $\mathcal{A}$ is not allowed to issue *Leak Queries* and *Update Queries*, and $\mathcal{A}$'s advantage is defined as $\mathrm{Adv}^{IND-CPA}_{VPEKS,\mathcal{A}}(1^\lambda) = |\Pr[b' = b]\text{-}1/2|$.

3.2 Construction

The basic idea underlying the construction is similar to that in Hu et al.'s method of transforming a continual master-key leakage-resilient anonymous IBE scheme to a continual leakage-resilient PEKS scheme. Specifically, we apply IBE-to-PEKS transformation in [1] to a continual master-key leakage-resilient anonymous IBE scheme. In this transformation, the cipher of keyword w includes (C, R) where $C \leftarrow$IBE.Enc(pk, w, R) and R is a random message. To enable it to verify the search result, we add to the keyword cipher a value computed by a pseudorandom function F taking the hash value of all the data ciphers associated with the search keyword as the key and taking a randomness extracted from w, R as its input, i.e. F(*Hash*(*allcipher*), *randExt*(w,R)) where randExt is a randomness extractor and $Hash(allcipher)$ is the hash value of all the data ciphers associated with the search keyword w. The cloud server should return data ciphers with their respective R and f=F(*Hash*(*allcipher*), *randExt*(w,R)). Then, the data user can compute f'=F(*Hash*(*allcipher*), *randExt*(w,R)) for each result itself and verify the completeness of the results by checking the equality of each f and f'. To allow the secret key update, the based IBE scheme should be equipped with key update algorithm as well. We call this transformation *IBE-to-VPEKS*.

Let $F(\cdot)$ be a pseudorandom function family and *randExt* be a randomness extractor. Let H be a hash function family. A VPEKS scheme (Setup, PEKS, Trapdoor, Test, Verify and UpdateKey) can be transformed from an anonymous IBE scheme with key-update(Setup, KeyGen, Encrypt, Decrypt, UpdateKey) as follows:

–VPEKS.Setup(λ): It runs $(pk, msk) \leftarrow$IBE.Setup(λ), and sets VPEKS.pk=pk,VPEKS.sk=msk.

–VPEKS.PEKS (pk,w): Let w be the keyword associated to a data file. It picks randomly R from the message space of IBE scheme and computes C=IBE.Encrypt(VPEKS.pk,w,R). Let $allcipher_w$ be all the data cipher associated with keyword w. Then, it computes $f = F(H(allcipher_w), randExt(w, R))$. The PEKS ciphertext CT of w is as (C, R, f).

–VPEKS.Trapdoor(sk,w): It runs $T_w \leftarrow$IBE.KeyGen(VPEKS.sk,w).

–VPEKS.Test(pk,CT,T_w): It parses CT as (C, R, f). If R=IBE.Decrypt(VPEKS.pk,T_w, C), it returns 1. Otherwise, it returns 0. After the search, the cloud server returns the related data cipher to the data user with R, f.

–VPEKS.Verify(Rst_w,w): Let $Rst_w = \{datacipher_i, R_i, f_i\}_{i=1\sim n}$ be the results of the search about keyword w returned by the cloud server, where n is the number of data associated with keyword w. The data user checks as follows:

Let $allcipher$ be all of the returned data ciphers;
Compute $h = H(allcipher)$;
For $i = 1$ to n do

$f_i' = F(h, randExt(w, R_i))$;

if $f_i' \neq f_i$ then the check fails, stop;
if all checks pass, then return true;

–VPEKS.UpdateKey(VPEKS.sk): It runs sk'=IBE.UpdateKey(VPEKS.sk,ϵ).

To prove the continual leakage security of above VPEKS scheme, we first show that the VPEKS scheme transformed by *IBE-to-VPEKS* is IND-CPA secure. Then, we prove that if the based anonymous IBE scheme is continual master-key leakage-resilient secure, then the VPEKS scheme constructed from it via *IBE-to-VPEKS* transformation is continual leakage-resilient secure.

Theorem 1 *Let IBE be an IBE scheme and let VPEKS be the verifiable PEKS scheme derived from IBE via IBE-to-VPEKS transformation. If IBE is ANO-IND-ID-CPA-secure, then VPEKS is IND-CPA-secure.*

Proof **Technique Outline.** The main difference between *IBE-to-VPEKS* and *IBE-to-PEKS* is that a pseudorandom value is added to the keyword cipher in *IBE-to-VPEKS*, i.e. f=F(*Hash*(*allcipher*), *randExt*(w,R)). To make it unable to improve the adversary's advantage, we change f to a true randomness step by step. At last, we can prove the security as same as that in [1]'s *IBE-to-PEKS* transformation.

We prove this theorem by a sequence of games as follows:

Game$_0$: is the IND-CPA security game of VPEKS.

Game$_1$: is the same as Game$_0$, except that f in the challenge cipher is changed with $f = F(H(allcipher_{w_b}), REXt)$ where $REXt$ is a randomness from the domain of the random extractor $randExt$.

Game$_2$: is the same as Game$_1$, except that f in the challenge cipher is changed with $RPfr$ where $RPfr$ is a randomness from the domain of the pseudorandom function $F(\cdot)$.

Game$_0$ and Game$_1$ are indistinguishable as $randExt(w_b, R)$ and $REXt$ are indistinguishable by the property of randomness extractors.

Game$_1$ and Game$_2$ are indistinguishable as f=F(H($allcipher_{w_b}$), $REXt$) and $RPfr$ are indistinguishable by the pseudorandomness of $F(\cdot)$.

Then we prove that the advantage of the adversary in Game$_2$ is negligible as follows which complete the proof.

Suppose $\mathcal{A}$ is an adversary in Game$_2$, we can construct an adversary $\mathcal{B}$ against the *ANO-IND-ID-CPA-secure* anonymous IBE scheme. $\mathcal{B}$ interacts with its own challenge $\mathcal{C}$ in *ANO-IND-ID-CPA-secure* game and plays the role of a challenger for $\mathcal{A}$ in Game$_2$ as below:

–Setup: Taking as input a security parameter 1^λ, the challenger $\mathcal{C}$ runs the IBE.Setup algorithm and sends the public key IBE.pk to $\mathcal{B}$. Then B sends IBE.pk to $\mathcal{A}$ as VPEKS public key VPEKS.pk. The challenger $\mathcal{C}$ retains the master secret key IBE.msk.

–Phase 1: Upon receiving the *Trapdoor* queries for the keyword w from $\mathcal{A}$, $\mathcal{B}$ responds as follow:

$\mathcal{B}$ takes the queried keyword w as"identity" and issues the *KeyGen Query*for the "identity" w to its own challenger $\mathcal{C}$. When $\mathcal{B}$ gets the secret key IBE.sk_w for the "identity" w, it forwards IBE.sk_w as trapdoor VPEKS.T_w to $\mathcal{A}$.

–Challenge: When $\mathcal{A}$ sends $\mathcal{B}$ two keywords w_0, w_1 where $|w_0| = |w_1|$, $\mathcal{B}$ constructs two message/identity pairs (w_0, R), (w_1, R) where R is a random message, and gives them to $\mathcal{C}$. When $\mathcal{B}$ receives its challenge ciphertext C=IBE.Encrypt(pk,w_β, R), it sends (C, R, $RPrf$) to $\mathcal{A}$ as VPEKS.CT, where $RPrf$ is a randomness from the domain of the pseudorandom function $F(\cdot)$.

–Phase 2: $\mathcal{A}$ can continue to issue *Trapdoor Queries* for any keyword w with the restriction that $w \neq w_0, w_1$.

–Guess: Finally, the adversary $\mathcal{B}$ outputs what the adversary $\mathcal{A}$ outputs.

We can get that $\mathcal{B}$ provides a perfect simulation for $\mathcal{A}$. Hence $\mathcal{B}$'s advantage against *ANO-IND-ID-CPA-secure* anonymous IBE scheme equals to the advantage of adversary in Game$_2$. Therefore, the advantage of adversary in Game$_2$ is negligible as the based anonymous IBE scheme is *ANO-IND-ID-CPA-secure*. □

The following theorem shows that if the based anonymous IBE scheme is continual master-key leakage-resilient secure, then the VPEKS scheme constructed from it via *IBE-to-VPEKS* transformation is continual leakage-resilient secure.

Theorem 2 *For the verifiable PEKS scheme constructed from the anonymous IBE scheme via IBE-to-VPEKS transformation, if the anonymous IBE scheme is* (l_{MSK}, l_{SK})*-masterkey-leakage anonymous, then the verifiable PEKS scheme is continual* l'_{SK}*-leakage secure under chosen plaintext attacks (*l'_{SK}*-cIND-CPA) where* l'_{SK}=l_{MSK}.

The proof of above theorem is as same as that of Theorem 5.2 in [28]. We omit it here. Also, we can get a concrete verifiable PEKS scheme secure against continual memory attacks by transforming the masterkey leakage anonymous IBE scheme presented in [28].

3.3 Extension to VPEKS-CML with leakage on key updates

We note that above definition of VPEKS-CML does not allow the leakage on key updates. To solve this problem, we extend above VPEKS scheme to one that allows leakage on key updates using the techniques introduced in [18]. In [18], the authors show how to compile any public-key encryption or signature scheme that satisfies a slight strengthening of CML which they call "consecutive" *Continual Leakage resilience* ("consecutive" CLR or 2CLR) without leakage on key updates to one that is continual leakage-resilient with leakage on key updates. Informally, the technique in [18] can be described as follow: Recall that the major obstacle to constructing a scheme which is secure against continual memory attacks with leakage on key updates is how to deal with leakage about the randomness used in key updates. In [18], they solve this problem by applying an *explainable compiler* to compile the key update algorithm. In this way, they can simulate leakage about the randomness with leakage about the input and the output of the key update algorithm. As the input and the output of the key update algorithm are in fact secret keys in successive two rounds, they require that the underlying scheme remains secure even when leakage about these two secret keys is available, i.e the scheme should be secure in the "consecutive" CLR model.

Here, we remark that the notion of "consecutive" CLR or 2CLR can be extended to IBE and VPEKS, and we omit the concrete 2CLR security definition of IBE and VPEKS. With the technique from [18], we can extend above VPEKS to one that is continual leakage-resilient with leakage on key updates. Specifically, we first show that the masterkey leakage anonymous IBE scheme presented in [28] is 2CLR secure. Then, a continual leakage-resilient IBE with leakage on key updates can be obtained by applying the compilation technique in [18]. At last, VPEKS scheme obtained as above is continual leakage secure with leakage on key updates.

We show that the masterkey leakage anonymous IBE scheme presented in [28] is 2CLR secure in the following theorem.

Theorem 3 *The anonymous IBE scheme presented in* [28] *is 2CLR secure.*

Proof Sketch Recall that security proof of the anonymous IBE scheme in [28] will generally observe the following process. First, the challenge ciphertext is replaced by a

semi-functional one. In this step, we need not consider leakage about secret keys as the indistinguishability between these two challenge ciphertexts holds even when the secret key is given. Next, the secret keys in the games are replaced semi-functional one by one. This is where we should deal with the leakage. Fortunately, when changing a secret key to a semi-functional one, the challenger has the ability to generate a normal secret key in the next round. So it will not break the indistinguishability if we face a stronger adversary who can obtain leakage about secret keys in successive two rounds. Finally, we prove the security in this setting. Therefore, with slight variation, original security proofs are still effective in proving the "consecutive" CLR security of the scheme in [28]. □

4 Extension to continual leakage-resilient verifiable dPEKS secure against keyword guessing attacks

In the original PEKS scheme, a secure channel between the server and the receiver should be assumed. Otherwise the linkability of PEKS cipher and a given trapdoor can be revealed to anyone. To solve this problem, Baek et al. [5] proposed a designated tester public-key encryption with keyword search (dPEKS) which inserts a mechanism into original PEKS and the communication does not rely on a secure channel. Also, an attacker can determine the trapdoor corresponds to which keyword by generating a PEKS ciphers for all the keywords and implementing the test algorithm for each cipher which is called "keyword guessing attacks" [41]. Recently, Chen [16] proposed a method of transforming anonymous IBE and traditional PKE to dPEKS secure against keyword guessing attacks which we call *(IBE,TE)-to-dPEKS*. In this section, we will extend the dPEKS definition to verifiable dPEKS (VdPEKS) and show that we can easily modify *(IBE,TE)-to-dPEKS* transformation to a new transformation called *(IBE,TE)-to-VdPEKS* which can transform an IBE scheme and a traditional PKE scheme to a verifiable dPEKS scheme. At last, we will prove that if the underlying IBE scheme is continual leakage-resilient secure, then the obtained VdPEKS scheme is continual leakage-resilient secure.

In the following, we will first describe the definition of verifiable dPEKS. Then we will present *(IBE,TE)-to-VdPEKS* transformation method and show the leakage-resilient security of the obtained VdPEKS scheme. Note that we equip VdPEKS with key update algorithm to get a continual leakage-resilient secure scheme.

4.1 VdPEKS definition

The VdPEKS definition can be obtained as follows by extending the definition of dPEKS introduced in [16]:

- GlobalSetup(λ): takes as input a security parameter λ, it outputs a global system parameter $\mathcal{SP}$ including a keyword space $\mathcal{W}$.
- KeyGen$_{Server}(\mathcal{SP})$: takes as input the system parameter $\mathcal{SP}$, it outputs a pair of public and secret keys, (pk_S, sk_S), of the server, S.
- KeyGen$_{Receiver}(\mathcal{SP})$: takes as input the system parameter $\mathcal{SP}$, it outputs a pair of public and secret keys, (pk_R, sk_R), of the receiver, R.
- dPEKS($\mathcal{SP}$, pk_R, pk_S, w): takes as input $\mathcal{SP}$, the receiver's public key, pk_R, the server's public key, pk_S, and a keyword, w. It returns a dPEKS ciphertext, C, of w.
- dTrapdoor($\mathcal{SP}$, pk_S, sk_R, w): takes as input $\mathcal{SP}$, the receiver's secret key, sk_R, the server's public key, pk_S, and a keyword, w. It outputs a trapdoor T_w.
- dTest($\mathcal{SP}$, C, sk_S, T_w): takes as input $\mathcal{SP}$, a dPEKS ciphertext, C, the server's secret key, sk_S, and a trapdoor, T_w. It outputs 1 if C and T_w correspond to the same keyword. Otherwise, it outputs 0.
- Verify(Rst_w,w): takes as input the result Rst_w and keyword w. It output 1 if the tester returns all the data associated with keyword w, and output 0 otherwise.
- UpdateKey(sk_R): takes as input the receiver's secret key, sk_R. It outputs a re-randomized key sk'_R, such that $|sk_R| = |sk'_R|$ and the distribution of sk_R is indistinguishable from the distribution of sk'_R.

The security definition of VdPEKS including *dPEKS indistinguishability against an adaptive chosen plaintext attack* and *trapdoor indistinguishability against an adaptive chosen plaintext attack* is as same as that in [41] which we omit here.

4.2 Construction

In this section, we show how to transform an IBE scheme and a traditional PKE scheme to a verifiable dPEKS scheme using *(IBE,TE)-to-dPEKS* transformation as follows.

–VdPEKS.GlobalSetup(λ): It sets $\mathcal{W}$ to be the ID space in IBE and returns the system parameter $\mathcal{SP}$.

–VdPEKS.KeyGen$_{Server}(\mathcal{SP})$: It takes $\mathcal{SP}$ to run (TE.pk, TE.sk)←TE.KeyGen of traditional PKE and generate the public and secret key pair VdPEKS.pk_S= TE.pk, VdPEKS.sk_S=TE.sk.

–VdPEKS.KeyGen$_{Receiver}(\mathcal{SP})$: It takes $\mathcal{SP}$ to run (IBE.pk, IBE.msk)←IBE.Setup, and sets VdPEKS.pk_R= IBE.pk, VdPEKS.sk_R= IBE.msk.

–VdPEKS.dPEKS($\mathcal{SP}$, pk_R, pk_S, w): Let w be the keyword associated to a data file and let $allcipher_w$ be all the data cipher associated with keyword w. It

picks randomly R from the message space of IBE scheme and computes the dPEKS ciphertext C of w as C=TE.Encrypt($\mathcal{SP}$, pk_S,CT), where CT=(IBE.Encrypt($\mathcal{SP}$, pk_R,w,R),R,f).

–VdPEKS.dTrapdoor($\mathcal{SP}$, pk_S, sk_R, w): It runs T_w ←TE.Encrypt($\mathcal{SP}$, pk_S,IBE.KeyGen($\mathcal{SP}$, sk_R,w)).

–VdPEKS.dTest($\mathcal{SP}$, C, sk_S, T_w): It first generates T'= TE.Decrypt($\mathcal{SP}$, sk_S, T_w) and CT'=TE.Decrypt($\mathcal{SP}$, sk_S, C). Then it parses CT' as (ct, R, f). If R= IBE.Decrypt($\mathcal{SP}$, T, ct), it returns 1. Otherwise, it returns 0. After the search, the cloud server returns the related data cipher to the data user with R, f.

–VdPEKS.Verify(Rst_w,w): Let $Rst_w = \{datacipher_i, R_i, f_i\}_{i=1\sim n}$ be the results of the search about keyword w returned by the cloud server, where n is the number of data associated with keyword w. The data user checks as follows:

Let $allcipher$ be all of the returned data ciphers;

Compute $h = H(allcipher)$;

For $i = 1$ to n do

$f_i' = F(h, randExt(w, R_i))$;

if $f_i' \neq f_i$ then the check fails, stop;

if all checks pass, then return true;

–VdPEKS.UpdateKey(sk_R): It runs sk_R'=IBE.UpdateKey(sk_R, ϵ).

Security In [16], the author proved that the dPEKS scheme obtained by applying *(IBE,TE)-to-dPEKS* satisfies *dPEKS indistinguishability against an adaptive chosen plaintext attack* and *trapdoor indistinguishability against an adaptive chosen plaintext attack*. Note that the GlobalSetup, KeyGen_{Server}, $\text{KeyGen}_{Receiver}$ and dTrapdoor algorithms in *(IBE,TE)-to-VdPEKS* are the same as that in *(IBE,TE)-to-dPEKS*. The key mechanism used in dTest algorithm in *(IBE,TE)-to-VdPEKS* is the same as that in *(IBE,TE)-to-dPEKS*. Similar to that of *IBE-to-VPEKS* and *IBE-to-PEKS*, the main difference between *(IBE,TE)-to-VdPEKS* and *(IBE,TE)-to-dPEKS* is that the output of dPEKS algorithm in *(IBE,TE)-to-VdPEKS* is obtained by encrypting CT using traditional PKE under pk_S where CT=(IBE.Encrypt($\mathcal{SP}$, pk_R,w,R),R,f), while CT=(IBE.Encrypt($\mathcal{SP}$, pk_R,w,R),R) in *(IBE,TE)-to-dPEKS*. Therefore, by applying the same techniques used in the proof of Theorem 3.3 to the proofs in [16], we can prove that the VdPEKS scheme transformed via *(IBE,TE)-to-VdPEKS* also satisfies *dPEKS indistinguishability against an adaptive chosen plaintext attack* and *trapdoor indistinguishability against an adaptive chosen plaintext attack*. The continual leakage-resilient security of above VdPEKS scheme can be proved using the same technique used in proofs of Theorem 2 and 3 which we omit here.

5 Conclusion

In this paper, we focus on how to construct verifiable public-key encryption with keyword search which enables users to search on encrypted data and efficiently verify the completeness of the search results and is secure against continual memory attacks on the trapdoor generation process. To this end, we propose *IBE-to-VPEKS* transformation and apply it to an continual masterkey leakage resilient secure anonymous IBE scheme. The obtained VPEKS scheme is proved to be secure against continual memory attacks. Besides, we extend our VPEKS scheme to be continual leakage resilient with leakage on key updates which make it resist more information leakage. To make the scheme secure against keyword guessing attacks, we extend *IBE-to-VPEKS* transformation to *IBE-to-VdPEKS* transformation and prove the security of the resulting VdPEKS scheme. As in our continual leakage secure construction with leakage on key updates, obfuscation is applied to obtain an obfuscated key update program in the key generation process, the limitation of our scheme is obvious, i.e., the key generation process is somewhat inefficient. Fortunately, the key generation process needs to be executed only once in the whole lifetime of the construction so that it does not reduce the effectiveness of the construction.

Acknowledgements This project is supported in part by National Natural Science Foundation of China (No.61602275, 61632020, 61602468, 61772311), Shandong Province Higher Educational Science and Technology Program (No.J15LN01), the Open Project of Co-Innovation Center for Information Supply & Assurance Technology, Anhui University(No.ADXXBZ201702).

References

1. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2005) Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions. In: Advances in cryptology-crypto 2005, pp 205–222
2. Akavia A, Goldwasser S, Vaikuntanathan V (2009) Simultaneous hardcore bits and cryptography against memory attacks. In: TCC 2009, pp 474–495
3. Ateniese G, Fu K, Green M, Hohenberger S (2006) Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans Inf Syst Secur 9(1):1–30
4. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D (2007) Provable data possession at untrusted stores. In: CCS 2007. ACM, pp 598–609
5. Baek J, Safiavi-Naini R, Susilo W (2008) Public key encryption with keyword search revisited. In: ICCSA 2008, pp 1249–1259
6. Ballard L, Kamara S, Monrose F (2005) Achieving efficient conjunctive keyword searches over encrypted data. In: ICICS 2005, pp 414–426
7. Bertino E, Paci F, Ferrini R, Shang N (2009) Privacy-preserving digital identity management for cloud computing. In: IEEE Data engineering bulletin, vol 32, pp 21–27

8. Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: Advances in Cryptology-CRYPTO 1997. Springer, pp 513–525
9. Biham E, Carmeli Y, Shamir A (2008) Bug attacks. In: Advances in cryptology-CRYPTO 2008. Springer, pp 221–240
10. Boneh D, Waters B (2007) Conjunctive, subset and range queries on encrypted data. In: TCC 2007, pp 535–554
11. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Advances in cryptology-eurocrypt 2004, pp 506–522
12. Brakerski Z, Kalai Y, Katz J, Vaikuntanathan V (2010) Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage. In: FOCS 2010, pp 501–510
13. Byun J, Rhee H, Park H, Lee D (2006) Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: SDM 2006, pp 75–83
14. Chen X, Huang X, Li J, Ma J, Wong D, Lou W (2015) New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Trans Inf Forens Secur 10(1):69–78
15. Chen X, Li J, Ma J, Weng J, Lou W (2016) Verifiable computation over large database with incremental updates. IEEE Trans Comput 65(10):3184–3195
16. Chen Y (2015) Speks: secure server-designation public key encryption with keyword search against keyword guessing attacks. Comput J 58(4):922–933
17. Chen Y, Zhang Z, Lin D, Cao Z (2012) Anonymous identity-based hash proof system and its applications. In: ProvSec 2012, pp 143–160
18. Dachman-Soled D, Dov Gordon S, Liu F, O'Neill A, Zhou H (2016) Leakage-resilient public-key encryption from obfuscation. In: PKC 2016, pp 101–128
19. Dodis Y, Kalai Y, Lovett S (2009) On cryptography with auxiliary input. In: STOC 2009, pp 621–630
20. Dziembowski S, Pietrzak K (2008) Leakage-resilient cryptography. In: FOCS 2008, pp 293–302
21. Fortis T, Munteanu V, Negru V (2015) A taxonomic view of cloud computing services. Int J Comput Sci Eng 11(1):17–28
22. Gandolfi K, Mourtel C, Olivier F (2001) Electromagnetic analysis: concrete results. In: CHES 2001, pp 251–261
23. Gao C, Cheng Q, He P, Susilo W, Li J (2018) Privacy-preserving naive bayes classifiers secure against the substitution-then-comparison attack. Inform Sci 444:72–88
24. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: STOC 2009. ACM, pp 169–178
25. Goh EJ (2003) Secure indexes cryptology. ArXiv:2003:216
26. Golle P, Staddon J, Waters B (2004) Secure conjunctive keyword search over encrypted data. In: ACNS 2004, pp 31–45
27. Halderman J, Schoen S, Nadia H, Clarkson W, Paul W, Calandrino J, Feldman A, Appelbaum J, Felten E (2008) Lest we remember: cold-boot attacks on encryption keys. In: USENIX security symposium 2008, pp 45–60
28. Hu C, Yang R, Liu P, Yu Z, Y Z XuQ (2016) Public-key encryption with keyword search secure against continual memory attacks. Secur Commun Netw 9(11):1613–1629
29. Joshi J, Bhatti R, Bertino E, Ghafoor A (2004) Access control language for multidomain environments. IEEE Internet Comput 8(6):40–50
30. Kocher P (1996) Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in Cryptology-CRYPTO 1996. Springer, pp 104–113
31. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Advances in Cryptology-CRYPTO 1999. Springer, pp 388–397
32. Lai J, Zhou X, Deng RH, Li Y, Chen K (2013) Expressive search on encrypted data. In: AisaCCS 2013, pp 243–252
33. Li H, Liu D, Dai Y, Luan T, Shen X (2015a) Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage. IEEE Transa Emerg Topics Comput 3(1):127–138
34. Li H, Yang Y, Dai Y, Yu S, Xiang Y (2017) Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. IEEE Transactions on Cloud Computing. https://doi.org/10.1109/TCC.2017.2769645
35. Li H, Liu D, Dai Y, Luan T, Yu S (2018a) Personalized search over encrypted data with efficient and secure updates in mobile clouds. IEEE Trans Emerg Topics Comput 6(1):97–109
36. Li J, Chen X, Xhafa F, Barolli L (2015b) Secure deduplication storage systems supporting keyword search. J Comput Syst Sci 81(8):1532–1541
37. Li J, Liu Z, Chen X, Tan X, Wong D (2015c) L-encdb: a lightweight framework for privacy-preserving data queries in cloud computing. Knowl-Based Syst 79:18–26
38. Li J, Li J, Xie D, Cai Z (2016) Secure auditing and deduplicating data in cloud. IEEE Trans Comput 65(8):2386–2396
39. Li J, Chen X, Chow S, Huang Q, Wong D, Liu Z (2018b) Multi-authority fine-grained access control with accountability and its application in cloud. J Netw Comput Appl 112:89–96
40. Li S, Cui J, Zhong H, He Q (2017b) Lepa: a lightweight and efficient public auditing scheme for cloud-assisted wireless body sensor networks. Secur Commun Netw 2017(11):1–16
41. Rhee H, Park J, Susilo W, Lee D (2010) Trapdoor security in a searchable public-key encryption scheme with a designated tester. J Syst Softw 83(5):763–771
42. Shamir A (1984) Identity based cryptosystems and signature schemes. In: Advances in Cryptology-CRYPTO 1984. Springer, pp 47–53
43. Shen J, Gui Z, Ji S, Shen J, Tan H, Tang Y (2018) Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks. J Netw Comput Appl 106:117–123
44. Song D, Wagner D, Perrig A (2000) Practical techniques for searching on encrypted data. In: Security & Privacy 2000, pp 44–55
45. Xu Y, Wang M, Zhong H, Cui J, Liu L, Franqueira V (2017) Verifiable public key encryption scheme with equality test in 5g networks. IEEE Access 5:12,702–12,713
46. Yu J, Ren K, Wang C, Varadharajan V (2015) Enabling cloud storage auditing with key-exposure resistance. IEEE Trans Inf Forens Secur 10(6):1167–1179
47. Zheng Q, Xu S, Ateniese G (2014) Vabks: verifiable attributebased keyword search over outsourced encrypted data. In: Infocom 2014, pp 522–530
48. Zhong H, Cui J, Shi R, Xia C (2016) Many-to-one homomorphic encryption scheme. Secur Commun Netw 9(10):1007–1015
49. Zhong H, Zhu W, Xu Y, Cui J (2018) Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage. Soft Comput 22:243–251
50. Zhu B, Zhu B, Ren K (2011) Peksrand: providing predicate privacy in public-key encryption with keyword search. In: ICC 2011, pp 1–6

MFCNet: Multimodal Feature Fusion Network for RGB-T Vehicle Density Estimation

Ling-Xiao Qin, Hong-Mei Sun, Xiao-Meng Duan, Cheng-Yue Che, and Rui-Sheng Jia,

***Abstract*—The basic task of vehicle density estimation is to use image information to estimate the distribution and quantity of vehicles within it. However, many previous methods only use the optical information in red–green–blue (RGB) images, which makes it difficult to effectively identify potential vehicles under poor light, strong reflections, and bad weather, resulting in unsatisfactory density estimation performance. To address these problems, we consider introducing thermal images to provide a richer source of information for the vehicle density estimation task, and propose a multimodal feature fusion network (MFCNet) for accurate RGB-Thermal (RGB-T) vehicle density estimation. First, multimodal features are cross-integrated through the attention-guided multiscale feature fusion coordination module (MFFC) to compensate for the limitations of single modal features. Following this, the edge feature calibration module (EFC) is utilized to correct the spatial misalignment regions between modalities. Subsequently, the adaptive deep fusion module (ADFM) is applied to further refine the features on the global scale and improve the intermodality correlation. Finally, the features of different stages are fused step by step to obtain the final fused feature, which is fed into a simple regression header to generate a pixel-level vehicle density map. Experimental results show that the GAME2 and root mean square error of the proposed method are reduced to 5.21 and 3.54 on the DroneVehicle dataset, respectively. Compared with existing vehicle density estimation methods, MFCNet achieves competitive accuracy and can be applied to the vehicle density estimation task in unconstrained scenarios. Our codes will be available at https://github.com/QLingX/MFCNet.**

***Index Terms*—Multimodal feature fusion, multiscale feature fusion coordination (MFFC), RGB-Thermal (RGB-T) images, vehicle density estimation.**

I. Introduction

THE TASK of vehicle density estimation involves determining both the quantity and spatial distribution of vehicles within an unconstrained scene by analyzing a given urban traffic video image. Performing accurate vehicle density estimation can provide strong support for urban traffic management, parking management, traffic safety monitoring, and environmental protection, and promote the development of cities in the direction of intelligence and sustainability. Previously, there has been many works on vehicle density estimation, although these works have addressed many challenging factors (e.g., dense objects [1], [10], [41], drastic scale variations [2], [9], and complex background interferences [3], [11]), there still exist many complex situations in practical application scenarios, such as under unfavorable illumination conditions or in adverse weather environments. Therefore, accurate vehicle density estimation in unconstrained scenarios is still a very challenging task.

Most existing methods utilize only the optical information in red–green–blue (RGB) images, and while RGB images can provide strong support for density estimation under bright illumination conditions, lighting variations, and bad weather disturbances make them less reliable in practical applications. With the development of infrared cameras and depth cameras, it is possible to acquire multimodal information, which provides a richer source of information for vehicle density estimation tasks. Compared with RGB images, depth images can complement vehicle structure and spatial location information, but they are sensitive to environmental conditions and are prone to distortion during severe occlusion or insufficient illumination. In contrast, thermal images are less dependent on light, and by capturing the temperature distribution of the vehicle surface, they can provide stable information under various lighting conditions and have strong penetration to fog and haze. In addition, infrared cameras cover a wide-area and are suitable for highways, parking lots and urban traffic networks that require a wide range of monitoring scenarios. Therefore, for the vehicle density estimation in unconstrained scenes, thermal images are more suitable as supplementary information to RGB images. Fig. 1 illustrates the comparison of several pairs of RGB images and corresponding thermal images in the DroneVehicle dataset [4]. As shown in Fig. 1, RGB images provide rich detail and spatial distribution information of vehicles, but are sensitive to lighting conditions and are nearly invisible in dark or low light environments. Thermal images can provide temperature distribution information within the scene in dark or low light conditions, but they usually have low resolution and cannot provide the same detail and clarity as RGB images. In summary, RGB and thermal images offer significant complementary benefits. Introducing RGB-Thermal (RGB-T) vehicle density estimation can combine the advantages of RGB images and thermal images, so as to improve the performance of density estimation. However, it is a challenge to effectively

Received 25 June 2024; revised 11 September 2024 and 11 October 2024; accepted 15 October 2024. Date of publication 18 October 2024; date of current version 6 February 2025. This work was supported in part by the Humanities and Social Science Fund of the Ministry of Education of the People's Republic of China under Grant 21YJAZH077. *(Corresponding authors: Hong-Mei Sun; Rui-Sheng Jia.)*

The authors are with the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China (e-mail: qinlingxiao@sdust.edu.cn; skd991925@sdust.edu.cn; duanxiaomeng@sdust.edu.cn; chechengyue@sdust.edu.cn; skd990551@sdust.edu.cn).

Digital Object Identifier 10.1109/JIOT.2024.3483175

Fig. 1. Comparison of RGB images and thermal images under different lighting conditions in the DroneVehicle dataset.

fuse the features of RGB images and thermal images. Many previous related methods have made many attempts, but they failed to fully explore and integrate the complementary information of these two types of images, so the effect of density estimation still needs to be improved.

Therefore, in order to solve the above problems, this article proposes a multimodal feature fusion network (MFCNet) for RGB-T vehicle density estimation. Specifically, in order to fully integrate the multiscale information of the two modalities, this article designs a multiscale feature fusion coordination module (MFFC) to establish spatial correlations and coordinate the interaction of multimodal complementary features. Based on this, an edge feature calibration module (EFC) is developed to alleviate the spatial misalignment caused by fusion and enhance the edge features to promote the learning of intermodal discrepancy areas. In order to fully mine the high-level features, an adaptive deep fusion module (ADFM) is designed as a feature fusion block with a multilevel self-attentive cross-structure, which flexibly integrates the multiscale information at the global context level. Finally, the fusion features of each stage and the features output from the backbone are fused step by step by the multilevel compensation fusion module (MCFM) to obtain the final features with multiscale information and attention to the edge-complementary regions and contextual relationships for regression of vehicle density maps. In addition, to generate accurate density maps in cases of blurry images, this article designs the local similarity-aware loss to facilitate feature alignment by locally minimizing the angle difference between features. Overall, the main contributions of this article are as follows.

1) This article proposes an MFCNet for RGB-T vehicle density estimation. By fusing the cross-modal multiscale features in stages, MFCNet can generate more accurate vehicle density maps in scenes, such as illumination changes and vehicle occlusion, which significantly improve the performance of density estimation.
2) The MFFC designed in this article fully fuses the multiscale information of the two modalities by coordinating the multimodal complementary feature interaction. In addition, the EFC aims to emphasize the attention to the vehicle misalignment region between different modalities and dynamically adjust the detailed features in the fusion edge region. Finally, ADFM fully mines high-level features, captures deep correlations, and improves the expressiveness and robustness of the final fusion features.
3) To meet the requirements of RGB-T vehicle density estimation task, we relabel the RGB-T vehicle detection dataset DroneVehicle. A large number of experiments are conducted on this dataset to prove the effectiveness of MFCNet and compare it with current state-of-the-art methods, and the experimental results demonstrate the superiority of MFCNet.

II. Related Work

A. Vehicle Density Estimation

Traditional vehicle density estimation mainly uses wireless sensors [5], microwave detectors [6], or radar [7], and other methods. However, in practical applications, these instruments are expensive, have limited coverage, provide incomplete vehicle monitoring data, and are not capable of real-time and accurate estimation of vehicle density within each scene. To address the problems of traditional methods, in recent years, the research on vehicle density estimation has begun to focus on methods based on computer vision. Lempitsky and Zisserman [8] creatively proposed to introduce the spatial information of the targets by learning the mapping between local features of images and their corresponding density map, which inspired the subsequent researchers to utilize the density map to capture the spatial distribution of vehicles. With the rapid development of deep learning technology, many density map-based methods have been proposed.

Zhang et al. [9] proposed a multicolumn convolutional network architecture (MCNN), which uses multiple parallel CNN columns to extract object features at different scales. Li et al. [10] proposed a pure convolutional network CSRNet that can understand highly congested scenes, which effectively expands the receptive field by introducing dilated convolution in the back-end of the network. Liu et al. [11] explored a framework that combines attention mechanism and multiscale deformable convolution (ADCrowdNet), which can adaptively focus on the object-dense regions, so as to better resist various background noises. Subsequently, Gao et al. [3] proposed a SCAR framework, which introduces a channel-wise attention module and a spatial-wise attention module to extract contextual and target attention information, so that the network pays more attention to the core regions relevant to the task and generates more accurate density maps. In addition, in order to improve the generalization ability of the model in practical applications and reduce the dependence on labeled data, some works [12], [13] explored adaptive object counting from the source domain to the target domain. With the application of vision transformer (ViT) [14] in the field of target counting, CCTrans [15], proposed by Tian et al., utilizes a pyramid feature aggregation model to aggregate semantic and detail information, and predicts the density maps through an efficient regression head with multiscale dilated convolution. Lin et al. [16] proposed a local learnable region attention network (MAN) to solve the problem that the fixed-size attention of transformers cannot cope well with

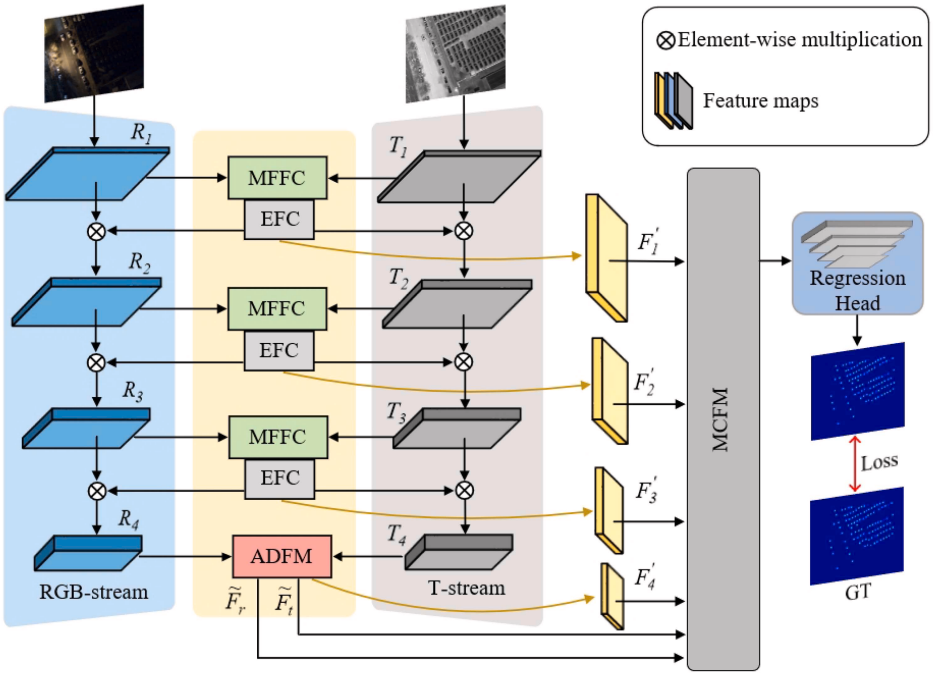


Fig. 2. Architecture of proposed MFCNet.

large-scale changes of objects in images and achieved good performance.

B. Multimodal Vehicle Density Estimation

The above studies mainly focus on exploiting the optical information in RGB images. However, only using RGB images as the information source may not be able to overcome challenges, such as illumination variations, occlusion problems, complex backgrounds, etc., and it is difficult to be applied to vehicle density estimation in unconstrained scenes. Therefore, vehicle density estimation based on multimodal becomes more essential. With the development and popularization of infrared cameras and depth cameras, RGB-T and RGB-Depth (RGB-D) vehicle density estimation have attracted widespread attention. The following is a brief review of these two multimodal vehicle density estimation methods.

1) RGB-D Methods: Compared with RGB images, depth images can provide additional vehicle structure and spatial location information, which can help to improve the performance of vehicle density estimation. Song et al. [17] first attempted to feed depth images into recurrent convolutional neural network (RCNN) for object counting aiming to improve the counting performance in crowded scenes. Lian et al. [18] proposed a regression guided detection network combining RGB information and depth information, which can accurately detect object bounding boxes by learning feature representations. Li et al. [19] made full use of multimodal complementary information through cross-modal cycle-attention fusion, and introduced a fine-coarse supervision strategy to better deal with large scale variations of objects. However, depth cameras are sensitive to environmental conditions, and images can be distorted in the case of severe occlusion or insufficient illumination, so the RGB-T methods are worth exploring.

2) RGB-T Methods: Peng et al. [20] constructed the first RGB-T crowd counting dataset based on drone-view and proposed the MMCCN network to learn multimodal features for object counting. Liu et al. [21] constructed a large-scale RGBT-CC benchmark and proposed a new cross-modal collaborative representation learning method to focus on the interaction and fusion of information from different modalities. Tang et al. [22] introduced a three-stream adaptive fusion network named TAFNet. The main stream is used to extract combination features, and two auxiliary streams are used to extract modality-specific features as auxiliary information, effectively utilizing both combination features and modal features. Subsequently, Zhou et al. [23] proposed a dual-branch enhanced feature fusion network to generate density maps with clear distribution of objects by integrating RGB-T fusion features of different scales. Li et al. [24] explored how to aggregate multiscale contextual features in the process of feature extraction and cross-modal feature fusion, and achieved good performance. In addition, there are some works, such as CAGNet [25] and GETANet [42], which also achieved advanced performance. However, these methods focus on extracting information from branches for additional fusion, and are somewhat lacking in the information interaction between branches, the global context and the processing of fusion edges. Therefore, this article adopts a new fusion strategy, which not only enhances the information interaction between branches but also provides a careful treatment of the fusion edge, so as to improve the effect of cross-modal feature fusion and achieve better density estimation performance.

III. Proposed Method

A. Method Overview

In this article, MFCNet is proposed for RGB-T vehicle density estimation. The network adopts a two-stream network architecture, whose input is the paired RGB image and thermal

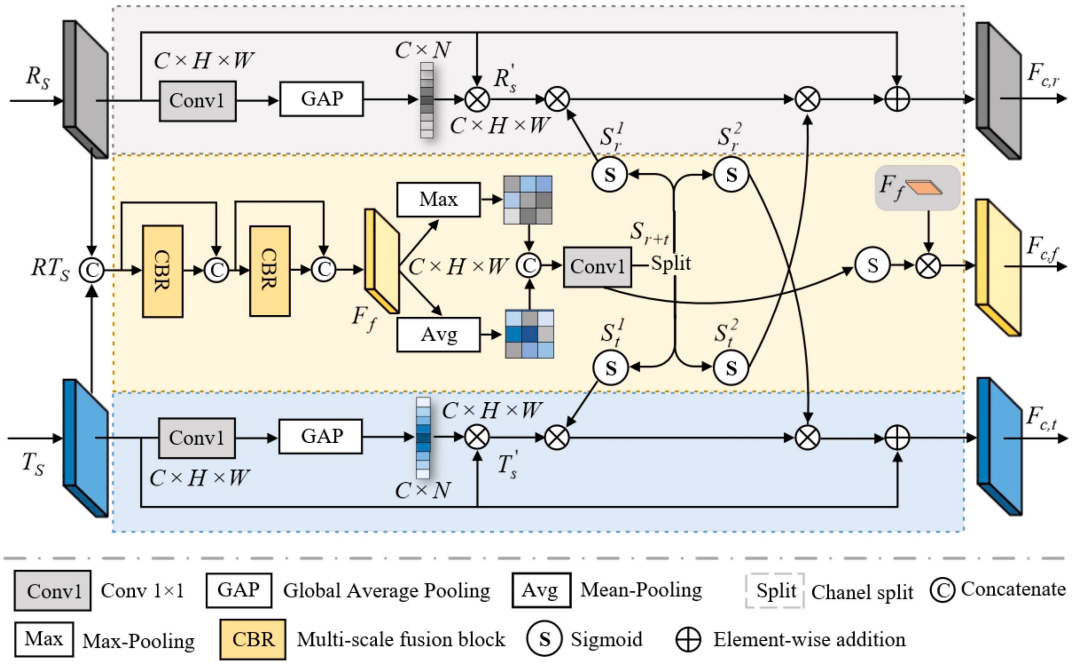


Fig. 3. Structure of proposed MFFC.

image, and output is the vehicle distribution density map predicted by the regression head. The network consists of four modules: 1) MFFC; 2) EFC; 3) ADFM; and 4) MCFM, as shown in Fig. 2. MFCNet uses the state-of-the-art ConvNext network [26] as the backbone to extract the RGB features and thermal features of the four stages. Specifically, MFFC fully integrates the multiscale information of the two modalities, establishes spatial correlations, and coordinates interactions of multimodal complementary features. On this basis, EFC aims to alleviate the spatial misalignment caused by fusion and enhance the edge features to promote the learning of intermodal discrepancy areas. ADFM is designed as a feature fusion block with a multilevel self-attentive cross-structure, which fully exploits the high-level features and flexibly integrates the multiscale information at the global context level. Finally, the fusion features of each stage and the multimodal features output from the backbone are fused step by step in the MCFM to obtain the final features with multiscale information and attention to the edge-complementary regions and contextual relationships. The details of the individual modules are provided in the following sections.

B. Multiscale Feature Fusion Coordination Module

RGB images can provide high-resolution information and rich color information, which is helpful for feature analysis in details, while thermal images provide excellent visibility in special environments. To overcome the limitation of a single modality, thermal images are introduced for RGB-T vehicle density estimation in this article. By fusing RGB images and thermal images, the network is able to fully utilize the complementary advantages of the two modalities, so that the density map output by the network can provide accurate vehicle distribution information even under adverse conditions.

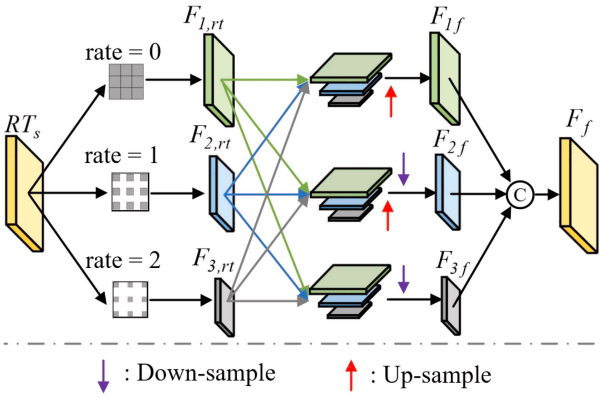


Fig. 4. Structure of proposed CBR.

In order to make the network more effective in fusing cross-modal information, the MFFC module is designed in this article.

As shown in Fig. 3, MFFC mainly consists of the multiscale fusion of RGB features and thermal features, and the subsequent feature coordination operation. In order to prevent multimodal features from disturbing each other and ensure spatial consistency for fusion, the convolution-BatchNorm-ReLU (CBR) module is designed in this article, as shown in Fig. 4. The CBR consists of three branches corresponding to three different dilation convolutions, which is designed to fuse features on multiple scales. Specifically, the features of the two modalities are fed into the CBR module after the concatenate operation. In the CBR module, the features are processed by dilated convolution with dilation rates 0–2 and summed to obtain the fused features $F_{\text{if}} \in \Re^{C \times H_i \times W_i} (i = 1, 2, 3)$ at different scales, respectively. Finally, these fused features

are concatenated and generate by 3×3 convolution with rich detail and spatially consistent multiscale fused feature $F_f \in \Re^{C\times H\times W}$. The above process is mainly expressed as follows:

$$RT_s = \text{Concat}(R_s, T_s), s = 1, 2, 3, 4 \quad (1)$$

$$F_{i,rt} = DC(RT_s, d = k), i = 1, 2, 3; k = 0, 1, 2 \quad (2)$$

$$F_{1f} = F_{1,rt} + UP^2(F_{2,rt}) + UP^4(F_{3,rt}) \quad (3)$$

$$F_{2f} = DW_2(F_{1,rt}) + F_{2,rt} + UP^2(F_{3,rt}) \quad (4)$$

$$F_{3f} = DW_4(F_{1,rt}) + DW_2(F_{2,rt}) + F_{3,rt} \quad (5)$$

$$F_f = \text{Conv}(\text{Concat}(F_{1f}, F_{2f}, F_{3f})) \quad (6)$$

where Concat(·) represents the concatenate operation, $DC(\cdot, d = k)$ represents a dilated convolution with a kernel size of 3×3, and k is the dilated rate. $UP^i(\cdot)$ represents the upsampling operation of i times, $DW_i(\cdot)$ represents the downsampling operation of i times, and Conv(·) represents a convolution with a kernel size of 3×3.

In order to improve the network's ability to perceive the fused key regions and enhance the expressiveness of the fused features, spatial attention is introduced after multiscale feature fusion to make the network focus on a specific local region, so as to better deal with the fusion details and improve the network's performance in complex scenes. Spatial attention consists of two parts: 1) max-pooling (MAX) and 2) mean-pooling (AVG). Specifically, the features obtained from these two operations are then combined to obtain the spatially rich multiscale fusion feature $F_{r+t} \in \Re^{C\times H\times W}$, which is then activated by Sigmoid and multiplied with the multiscale fusion feature F_f to obtain the final output of the fusion part $F_{c,f} \in \Re^{C\times H\times W}$. The above process can be expressed as

$$F_{r+t} = \text{Conv}(\text{Concat}(\text{MAX}(\text{Conv}(F_f)) + \text{AVG}(\text{Conv}(F_f)))) \quad (7)$$

$$F_{c,f} = \text{Sig}(F_{r+t}) \odot F_f \quad (8)$$

where Sig(·) represents the sigmoid activation function.

Currently, many networks have made significant progress in multimodal fusion, but they usually focus on the fusion between different modalities and neglect the deep mining of the modalities themselves. For this reason, channel attention is introduced in this article to strengthen the characteristics of each modality itself. Channel attention extracts the global information of each channel through global average pooling (GAP), which enables the network to mine modal key features more effectively and reduce the interference of redundant information. Meanwhile, feature crossover facilitates the transfer and fusion of intermodal information, further enhancing the utilization of complementary modal information. Specifically, the features of the two modalities are first subjected to GAP operation separately to obtain the representative mean value of each channel, and multiplied with the original features to obtain the modality-enhancing features $R'_s \in \Re^{C\times H\times W}$ and $T'_s \in \Re^{C\times H\times W}$ to refine the feature representation. For the multiscale fusion feature F_{r+t}, channel split is applied to it and then fed into the sigmoid activation function to obtain the attention maps $S^1_r \in \Re^{C\times H\times W}$, $S^2_r \in \Re^{C\times H\times W}$, $S^1_t \in \Re^{C\times H\times W}$, and $S^2_t \in \Re^{C\times H\times W}$ that are biased toward each modality.

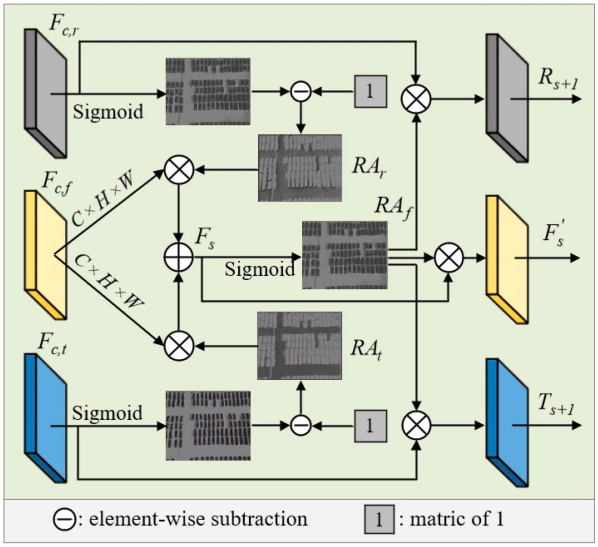


Fig. 5. Structure of proposed EFC.

Among them, S^1_r and S^1_t are multiplied with their respective modality-enhanced features R'_s and R'_t to refine their own features, and the enhanced features $F_{rs} \in \Re^{C\times H\times W}$ and $F_{ts} \in \Re^{C\times H\times W}$ that supplement the fusion information are obtained. S^2_r and S^2_t are multiplied by F_{ts} and F_{rs}, respectively to cross-fuse the key information emphasized by the other modality to achieve the purpose of coordinating cross-modal complementary information. Finally, the fused features of each modality are summed with the original features to obtain the final outputs $F_{c,r} \in \Re^{C\times H\times W}$ and $F_{c,t} \in \Re^{C\times H\times W}$ of MFFC. The above process can be expressed as

$$\begin{cases} R'_s = \text{GAP}(\text{Conv}(R_s)) \odot R_s \\ T'_s = \text{GAP}(\text{Conv}(T_s)) \odot T_s \end{cases} \quad (9)$$

$$S^i_r, S^i_t = \text{Sig}(\text{split}(\text{Conv}(F_{r+t}))), i = 1, 2 \quad (10)$$

$$\begin{cases} F_{rs} = S^1_r \odot R'_s \\ F_{ts} = S^1_t \odot T'_s \end{cases} \quad (11)$$

$$\begin{cases} F_{c,r} = (F_{rs} \odot S^2_t) + R_s \\ F_{c,t} = (F_{ts} \odot S^2_r) + T_s \end{cases} \quad (12)$$

where split(·) represents the channel splitting operation.

C. Edge Feature Calibration Module

Since it is difficult to completely synchronize the RGB images and thermal images in the process of collection, the misalignment of vehicle positions in the same scene across different modalities is inevitable. Such misalignment will lead to information loss and feature confusion during the fusion process, thereby reducing the quality of the fusion feature. To address this problem, inspired by the reverse attention mechanism [27], this article designs the EFC module, which aims to emphasize the attention to the vehicle misalignment region between different modalities and dynamically adjust the detailed features of the fusion edge region. The structure of the EFC is shown in Fig. 5.

Specifically, EFC receives $F_{c,r}$ and $F_{c,t}$ and the fusion feature $F_{c,f}$ which are outputted by MFFC. First, a sigmoid

activation function is applied to $F_{c,r}$ and $F_{c,t}$ to obtain the attention maps emphasizing the vehicle region, and then subtracted them from 1 to obtain the reverse attention maps $RA_r \in \Re^{C\times H\times W}$ and $RA_t \in \Re^{C\times H\times W}$ which emphasize the background region. This process is formulated as follows:

$$RA_{\mathrm{i}} = 1 - \mathrm{Sig}\left(F_{c,\mathrm{i}}\right), \mathrm{i} = r, t. \tag{13}$$

In order to make full use of the complementary information of the two modalities and strengthen the attention to the edge region, RA_r and RA_t are multiplied with the fusion feature $F_{c,f}$, respectively, and then summed up to obtain the fusion feature $F_s \in \Re^{C\times H\times W}$ which focuses on the edge information. Subsequently, a sigmoid activation function is applied to F_s to obtain the attention map $RA_f \in \Re^{C\times H\times W}$ that emphasizes the spatial misalignment region and edge information. Finally, RA_f is multiplied by $F_{c,r}$ and $F_{c,t}$, respectively, to calibrate the misalignment regions between modalities and retain the original features, while multiplying by F_s to obtain the fusion feature $F'_s \in \Re^{C\times H\times W}$. The formula is expressed as follows:

$$F_s = \left(F_{c,f} \odot RA_r\right) + \left(F_{c,f} \odot RA_t\right), s = 1, 2, 3 \tag{14}$$

$$RA_f = \mathrm{Sig}(F_s) \tag{15}$$

$$F'_s = F_s \odot RA_f \tag{16}$$

$$\begin{cases} R_{s+1} = F_{c,r} \odot RA_f \\ T_{s+1} = F_{c,t} \odot RA_f. \end{cases} \tag{17}$$

D. Adaptive Deep Fusion Module

In the previous sections, we explored the effectiveness of MFFC module and EFC module in local feature fusion. However, in the later stages of the network, a deep understanding of the global context becomes particularly important, especially when dealing with complex multiscale scenarios. Therefore, we design the ADFM module in the fourth stage of the network, which utilizes the multihead self-attention mechanism to deepen the intermodal connections through long-distance multimodal feature interactions, thus enhancing the depth and expressiveness of feature fusion. The detailed structure of the ADFM module is shown in Fig. 6.

Specifically, the RGB feature $R_4 \in \Re^{C\times H\times W}$ and the thermal feature $T_4 \in \Re^{C\times H\times W}$ are first processed by the SAFE module to enhance the features of each modality. SAFE adopts a multihead self-attention architecture and uses SR to reduce the computational cost. The formula is expressed as follows:

$$\begin{cases} R'_4 = SAFE(R_4) \\ T'_4 = SAFE(T_4). \end{cases} \tag{18}$$

Then, the enhanced features $R'_4 \in \Re^{C\times H\times W}$ and $T'_4 \in \Re^{C\times H\times W}$ are transformed into query, key and value vectors through the linear layer. In order to maximize the fusion of global context information of the other modality, we only retain the query of each modality in the self-attention calculation, while key and value are replaced by the fusion vectors using the corresponding vectors of the two modalities spliced together. Benefiting from the excellent performance of the self-attention mechanism in handling long-range dependencies, the attention maps calculated by the key and the value of different modalities can achieve maximum feature alignment and cross-modal information interaction. Then, the attention maps $F_{a,r} \in \Re^{C\times H\times W}$ and $F_{a,t} \in \Re^{C\times H\times W}$ of two modalities are added together and fed into the MLP to capture the deep fusion features, so as to further improve the network's ability of recognizing the key information about vehicles. Finally, the output of MLP is multiplied with the enhanced features R'_4 and T'_4, respectively, to obtain the RGB features $\tilde{F}_r \in \Re^{C\times H\times W}$ and the thermal features $\tilde{F}_t \in \Re^{C\times H\times W}$ with cross-modal long-range complementary dependencies. The specific process is expressed as follows:

$$\begin{cases} C_q^r, C_k^r, C_v^r = \mathrm{Linear}\left(R'_4\right) \\ C_q^t, C_k^t, C_v^t = \mathrm{Linear}\left(T'_4\right) \end{cases} \tag{19}$$

$$C_j = \mathrm{Rsp}\left(\mathrm{Conv}\left(\mathrm{Concat}\left(\mathrm{Rsp}\left(F_j^r\right), \mathrm{Rsp}\left(F_j^t\right)\right)\right)\right), j = k, v \tag{20}$$

$$\begin{cases} F_{a,r} = \mathrm{Rsp}\left(\mathrm{Softmax}\left(C_q^r \odot C_k\right) \odot C_v\right) + R'_4 \\ F_{a,t} = \mathrm{Rsp}\left(\mathrm{Softmax}\left(C_q^t \odot C_k\right) \odot C_v\right) + T'_4 \end{cases} \tag{21}$$

$$\begin{cases} \widetilde{F}_r = \mathrm{MLP}\left(\left(F_{a,r} + F_{a,t}\right)\right) \odot R'_4 \\ \widetilde{F}_t = \mathrm{MLP}\left(\left(F_{a,r} + F_{a,t}\right)\right) \odot T'_4 \end{cases} \tag{22}$$

$$F'_4 = \mathrm{MLP}\left(\left(F_{a,r} + F_{a,t}\right)\right) \odot \left(F_{a,r} + F_{a,t}\right) \tag{23}$$

where $C_q^i \in \Re^{C\times N}$, $C_k^i \in \Re^{C\times N}$, and $C_v^i \in \Re^{C\times N}$ represent the query, key, and value vectors of the two modalities, $\mathrm{Rsp}(\cdot)$ represents the reshape operation, $C_j \in \Re^{C\times N}$ represents the key and the value after vector fusion, and $\mathrm{MLP}(\cdot)$ represents the multilayer perceptron.

E. Multilevel Compensation Fusion Module

The features outputted at the later stage of the network have high-level semantic information, while the features outputted at the earlier stage provide rich detail information. In order to gradually fuse the features at different levels and make full use of the feature analysis at the higher level and the detail supplementation ability at the lower level, this article designs the MCFM module, as shown in Fig. 7. The module adopts a multilevel multiscale fusion strategy, which effectively enhances the expression ability of the features by gradually connecting the fusion features of neighboring stages, so that the predicted density map output from network is more accurate and convergent. Specifically, each level of the multilevel fusion strategy consists of multiple CConv modules, and (24) demonstrates its basic operation. In order to prevent the loss of critical information due to multiple convolution operations, the network employs a sigmoid activation and feature weighting mechanism to generate features $F_{\mathrm{high}} \in \Re^{C\times H\times W}$, which reinforces high-level semantic information, and $F_{\mathrm{low}} \in \Re^{C\times H\times W}$, which is enriched with fine-grained information, respectively. Eventually, the two are fused to obtain the final fused feature $F_{fn} \in \Re^{2C\times H\times W}$, which simultaneously contains deep semantic information and focuses on the local detailed information. The specific process is represented as follows:

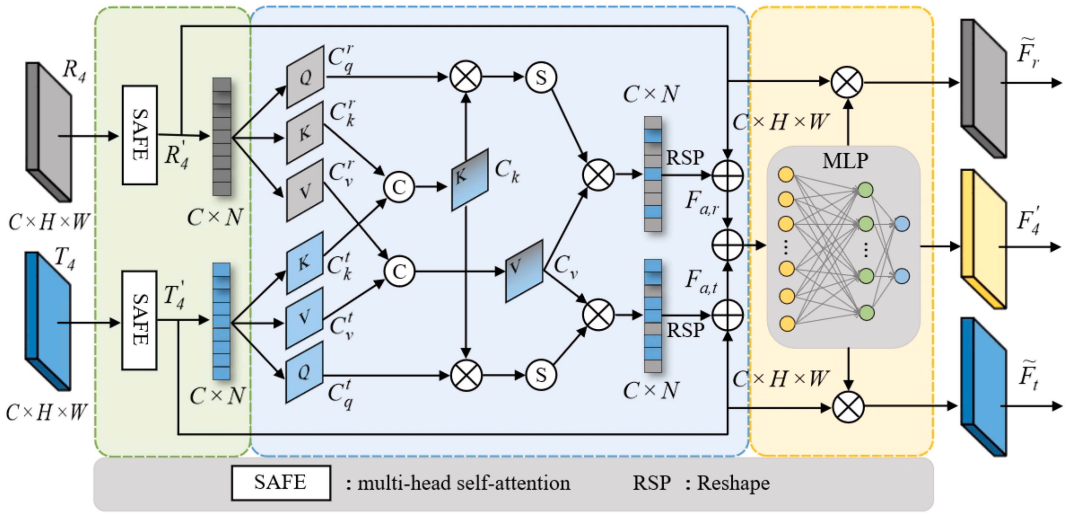


Fig. 6. Structure of proposed ADFM.

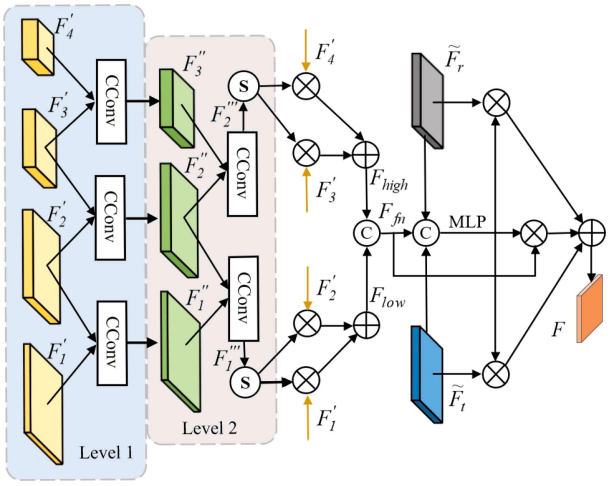


Fig. 7. Structure of proposed MCFM.

$$G_i' = \text{Concat(Relu)}(BN(\text{Conv}(G_i)))$$
$$\text{Relu}(BN(\text{Conv}(G_{i+1}))) \tag{24}$$
$$F_i'' = \text{CConv}(F_i', F_{i+1}'), i = 1, 2, 3 \tag{25}$$
$$F_i''' = \text{CConv}(F_i'', F_{i+1}''), i = 1, 2 \tag{26}$$
$$\begin{cases} F_{\text{high}} = \left(\text{Sig}(F_2''') \times F_4'\right) + \left(\text{Sig}(F_2''') \times F_3'\right) \\ F_{\text{low}} = \left(\text{Sig}(F_1''') \times F_2'\right) + \left(\text{Sig}(F_1''') \times F_1'\right) \end{cases} \tag{27}$$
$$F_{fn} = \text{Concat}(F_{\text{high}}, F_{\text{low}}). \tag{28}$$

In order to further refine the texture information and temperature distribution information of the fusion feature F_{fn}, this article introduces the RGB feature $\tilde{F}_r$ and the thermal feature $\tilde{F}_t$ output from the fourth stage of the network to make the final fusion with F_{fn}. First, $\tilde{F}_r$, $\tilde{F}_t$, and F_{fn} are preliminarily concatenated by concatenate operation, and subsequently the concatenated results are fed into the MLP module to further refine the features. Finally, the processed features are multiplied with the original features, respectively, and then summed up to integrate the multilevel and multiscale multimodal feature information to obtain the final feature $F \in \Re^{C\times H\times W}$ for regression density maps. This process can be expressed as follows:

$$F_m = \text{MLP}\left(\text{Concat}(\text{Conv}(\tilde{F}_r), \text{Conv}(\tilde{F}_t), \text{Conv}(F_{fn}))\right) \tag{29}$$
$$F = \left(F_m \times \tilde{F}_r\right) + \left(F_m \times \tilde{F}_t\right) + \left(F_m \times F_{fn}\right). \tag{30}$$

F. Loss Function

Cosine similarity loss is usually used to measure the similarity of different features in orientation, which promotes feature alignment and enhances information fusion between modalities by minimizing the angular difference between features at the global level. However, since the cosine similarity loss tends to focus on the overall similarity of feature vectors, the most significant feature has the highest attention, while the edge features are easily ignored, which results in poor performance of the network in the recognition of fuzzy features. To address this problem, this article designs the local similarity-aware loss L_{lc} based on the cosine similarity loss. Specifically, the feature map is divided into N patches, the feature similarity between the two modalities is calculated on each patch, and finally the average of all patches is used as the overall similarity. This ensures that the features within each patch are given sufficient attention, and helps the network to generate accurate and coherent density maps even in cases of high vehicle density or blurry input images. The definition of L_{lc} is as follows:

$$L_{lc} = \frac{1}{N}\sum_{i=1}^{N} |1 - \frac{F_{vi} \cdot F_{ir}}{||F_{vi}|| \cdot ||F_{ir}||}| \tag{31}$$

where N represents the number of patches, F_{vi} represents the feature vector of RGB image, and F_{ir} represents the feature vector of thermal image.

In order to better measure the difference between the density map generated by MFCNet and the ground-truth density map, inspired by [28], we use the weighted sum of optimal

transport (OT) loss and total variation (TV) loss as the density distribution loss L_{den}, which is defined as follows:

$$L_{\text{den}} = L_{OT}(D_i, D_j) + \lambda L_{TV}(D_i, D_j) \tag{32}$$

where D_i represents the predicted density map, D_j represents the ground-truth density map, and λ represents the loss factor, which is set to 0.1 as [28].

The total loss L of MFCNet is designed as a weighted sum of L_{lc}, L_{den}, and the counting loss L_{pred}, which is defined as follows:

$$L = L_{\text{pred}} + \alpha L_{\text{den}} + \beta L_{lc} \tag{33}$$

where α and β represent the loss factor for the density distribution loss L_{den} and the local similarity-aware loss L_{lc}, which are set to 0.2 and 0.1 and will be verified in Section IV.

IV. Experiments and Results

A. Experimental Setup

1) Dataset: We conducted extensive experiments on the DroneVehicle dataset [4] to evaluate the performance of the proposed MFCNet. The dataset was captured by a camera-equipped drone and covered a wide range of scenarios from day to night, such as urban roads, residential areas, parking lots, etc., with vehicle types, including cars, trucks, buses, and trains. It contains 28 439 RGB-T image pairs with the image resolution of 840×712. The DroneVehicle dataset originally used for the target detection task, which provides samples of oriented bounding box annotations. To accommodate the RGB-T vehicle density estimation task, we have relabeled this dataset to provide point-level annotations of vehicles to represent vehicle locations. In this article, 3000 images from the relabeled dataset are used for training, 800 for validation and 800 for testing.

2) Implementation Details: AdamW [29] is used to optimize the training. The learning rate is set to 5×10^{-5} and reduced it by a factor of 0.5 every 50 stages. The training period is set to 300 with the batch size of 4. In addition, we use random flipping with a probability of 0.5 on the dataset for data enhancement during training. During testing, the original images are fed into the network in their entirety to generate the final density maps. The experiments are all conducted in the same environment with the parameters shown in Table I.

3) Evaluation Metrics: In this article, the root mean-square error (RMSE) and the grid averaged mean absolute error (GAME) are used as evaluation metrics for the network. RMSE reflects the density estimation performance of the proposed MFCNet, but does not consider whether the vehicle location information is accurate. Therefore, the GAME evaluation metric is introduced to evaluate the localization accuracy of vehicle density estimation. RMSE and GAME are defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{\text{i}=1}^{N}\left(S^i - GT^i\right)^2} \tag{34}$$

$$\text{GAME}(L) = \frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{4^L}|S_n^l - GT_n^l| \tag{35}$$

TABLE I
Experimental Environment

Name	Parameter
System	Windows 10
Frame	Pytorch
Language	Python
CPU	Intel(R) Core (TM) i7-10870 CPU @ 2.50GHz
GPU	NVIDIA GeForce RTX 3060Ti
RAM	16.00 GB

where N represents the number of input images. S^i and GT^i represent the predicted vehicle density and the ground-truth vehicle density in the ith image, respectively. S_n^l and GT_n^l represent the estimated vehicle density and the ground-truth vehicle density for the lth region of the nth image. GAME first divides the image into a grid of 4^L nonoverlapping regions, and then calculates the sum of mean absolute error (MAE) of these regions, with the higher L representing the more stringent metric criteria, and GAME (0) is equal to MAE.

B. Experiment Results

1) Tests on the Dataset: In order to evaluate the density estimation performance of MFCNet when the vehicle scale varies and the illumination conditions are limited, we performed tests on the DroneVehicle dataset and visualized the test results, as shown in Fig. 8. The visualization results include RGB-T vehicle image pairs, corresponding ground truth, predicted vehicle density maps, and results from comparison experiments. The test results show that MFCNet has superior performance in density estimation, and the error between the prediction and ground-truth number of vehicles is small, with an average error of less than 4%. In addition, the high-resolution density maps generated by MFCNet provide detailed information on vehicle distribution, which is of great practical value. Therefore, the proposed MFCNet can be effectively applied to vehicle density estimation in unconstrained scenarios.

2) Comparison With State-of-the-Art Methods: In order to demonstrate the advancement of MFCNet in multimodal vehicle density estimation task, our method is compared with current state-of-the-art multimodal methods, including RGB-D methods and RGB-T methods. Among them, RGB-D methods include UCNet [30], BBSNet [31], and HDFNet [32]. RGB-T methods include MMCCN [20], CSRNet+IADM [10], SANet+IADM [2], BL+IADM [33], TAFNet [22], DEFNet [23], MC^{3}Net [34], CCRMNet [35], CSA-Net [24], and CAGNet [25]. In particular, in order to make a fair comparison, these comparison methods are re-evaluated on the DroneVehicle dataset, and the same training parameters are used to ensure the fairness of the comparison. The comparison results are shown in Table II, where the optimal results are shown in bold and the suboptimal results are shown in red.

We have selected vehicle images covering different scenarios and density levels for visualization, as shown in Fig. 8. It can be seen that the prediction results of MFCNet in various scenarios are closer to the number of vehicles in

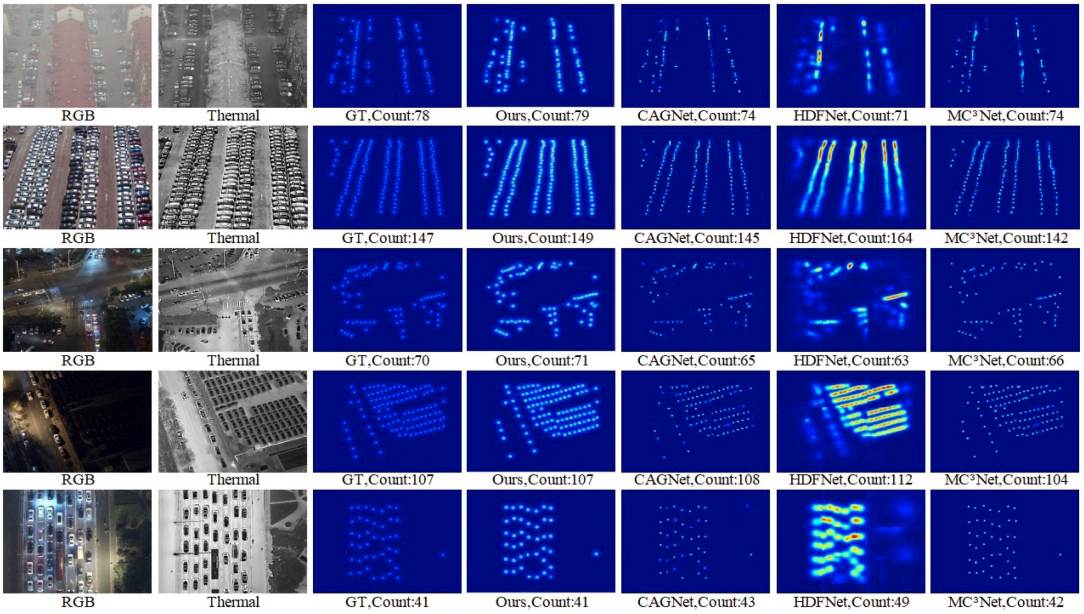


Fig. 8. Visualization of vehicle density maps generated for different scenarios.

TABLE II
COMPARISON RESULTS OF MFCNET WITH OTHER STATE-OF-THE-ART METHODS ON THE DRONEVEHICLE DATASET

Method	Venue&Year	GAME(0)↓	GAME(1)↓	GAME(2)↓	GAME(3)↓	RMSE↓
UCNet [30]	CVPR2020	10.77	13.68	16.72	21.44	12.17
BBSNet [31]	ECCV2020	8.25	9.86	13.15	17.88	11.45
HDFNet [32]	RESE2021	6.88	8.27	10.83	13.72	9.12
MMCCN [20]	ACCV2020	4.87	6.08	9.87	13.72	6.36
CSRNet+IADM [10]	CVPR2021	12.21	14.67	18.44	23.17	14.72
SANet+IADM [2]	CVPR2021	13.05	15.44	19.24	25.05	15.34
BL+IADM [33]	CVPR2021	12.71	15.28	18.74	24.23	14.85
TAFNet [22]	ISCAS2022	3.84	4.65	7.38	9.93	4.98
DEFNet [23]	T-ITS2022	4.57	5.51	8.17	11.27	5.81
MC^{3}Net [34]	T-ITS2023	2.88	4.05	6.26	9.65	4.17
CCRMNet [35]	ICCEA2023	4.19	5.37	8.34	11.87	5.23
CSA-Net [24]	ESWA2023	5.07	6.52	9.71	12.72	6.79
CAGNet [25]	ESWA2023	3.11	4.23	6.47	9.56	4.43
Ours	-	**2.42**	**3.63**	**5.21**	**8.31**	**3.54**

TABLE III
COMPARISON RESULTS OF MFCNET WITH OTHER STATE-OF-THE-ART METHODS USING ONLY SINGLE MODAL DATA AS INPUT

Method	Input Data	Venue&Year	GAME(0)↓	GAME(1)↓	GAME(2)↓	GAME(3)↓	RMSE↓
AMRNet [36]	RGB	ECCV2020	16.25	19.92	24.65	28.59	20.89
SASNet [37]		AAAI2021	18.93	22.18	29.21	34.03	23.51
GL [38]		CVPR2021	20.57	24.98	30.41	35.95	26.72
MAN [16]		CVPR2022	17.85	20.75	25.30	30.08	21.68
STNet [39]		TMM2022	14.89	17.24	21.62	25.11	17.45
MISC [40]		TIP2023	17.52	20.83	24.76	29.76	21.57
AMRNet [36]	Thermal	ECCV2020	10.08	11.32	14.63	19.17	11.24
SASNet [37]		AAAI2021	9.51	10.83	13.89	17.51	10.85
GL [38]		CVPR2021	9.95	11.22	14.19	18.21	11.61
MAN [16]		CVPR2022	11.27	13.11	16.20	19.72	13.59
STNet [39]		TMM2022	9.53	10.72	13.47	16.85	10.78
MISC [40]		TIP2023	8.93	9.23	12.55	15.72	9.55
Ours	RGB-T	-	**2.42**	**3.63**	**5.21**	**8.31**	**3.54**

ground truth than other methods. In high-density scenarios, MFCNet is able to significantly reduce the prediction error and effectively minimize the underestimation and overestimation of the number of vehicles. In low-density scenarios, even with the presence of more interfering objects with shapes similar to vehicles, MFCNet can still accurately estimate the vehicle

density, demonstrating its robustness in dealing with complex backgrounds. Table II gives the comparison results of MFCNet with other methods. It can be clearly seen that our method achieves very good results in all metrics. The GAME (2) and RMSE of MFCNet have been reduced by 16.77% and 15.11%, respectively, compared to the suboptimal method, MC^{3}Net. Compared with TAFNet, a state-of-the-art method that uses the same backbone network as MFCNet, the GAME (2) and RMSE of MFCNet have been reduced by 29.40% and 28.92%, respectively, further demonstrating the significant advantages of MFCNet, in terms of density estimation accuracy.

In addition, to further validate the superiority of MFCNet as a multimodal method, we compared the density estimation accuracy with the methods using single modal data as input, and the experimental results are shown in Table III. It can be seen that, compared to the methods using only RGB images or thermal images as input, MFCNet achieves satisfactory performance. In particular, the RMSE metric of MFCNet, under the condition of inputting only RGB images and thermal images, is reduced by 79.71% and 62.93%, respectively, compared with the current STNet and MISC method with the highest accuracy. This stems from the fact that single modal methods are usually limited when facing density estimation in complex scenes, whereas multimodal methods can utilize the complementary information of RGB and thermal modalities at the same time, thus overcoming the limitations of single modal methods.

3) Comparison of Performance of Predicted Density Maps: In order to investigated the quality of the predicted density maps generated by MFCNet, we compared them with the density maps generated by state-of-the-art methods under low-light conditions. As shown in Fig. 9(a), both MC^{3}Net and HDFNet suffer from different degrees of misclassification or blurred localization in occluded or blurred scenes, and these regions are marked with red boxes. In contrast, the density map generated by MFCNet is not only clear and accurate but also can accurately locate the position of the real vehicle, even at the edge of the image or for the occluded vehicle. In addition, we investigated the distribution of the density estimation results, and the visualization results are shown in Fig. 9(b). Each point in the figure represents a vehicle image sample, and the denser plot on the diagonal line represents that the network has a higher density estimation performance. It can be seen that MFCNet has the least outliers and almost all points are clustered on the diagonal, which indicates that the network possesses good fitting results and generalization ability.

4) Comparison of Performance Under Different Illumination Conditions: In order to further test the robustness of MFCNet to different lighting conditions, we divided the test dataset into bright images and dark images according to the light level to verify the performance of MFCNet under bright and dark conditions. Table IV lists the performance of MFCNet and the comparison results with other current state-of-the-art methods. The results show that MFCNet achieves better accuracy under both different conditions compared to other methods. In particular, the density estimation accuracy of the other methods under dark conditions are degraded to varying degrees compared with that of bright conditions, while MFCNet shows stable performance. This is attributed to the fact that MFCNet is able to efficiently capture the vehicle distribution information in thermal images to compensate the detail information lost in RGB images due to the low-light, thus maintaining a satisfactory performance.

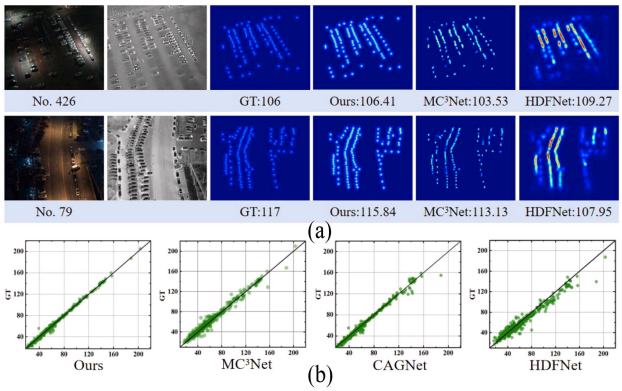


Fig. 9. Visualization of the comparison for density estimation details.

TABLE IV
COMPARISON RESULTS OF MFCNET WITH OTHER STATE-OF-THE-ART METHODS UNDER DIFFERENT LIGHTING CONDITIONS

Scenes	Method	GAME(0)↓	GAME(2)↓	RMSE↓
Bright	TAFNet [22]	4.28	8.89	6.03
	CCRMNet [35]	5.07	9.31	6.96
	CAGNet [25]	4.05	8.24	5.93
	MC^{3}Net [34]	3.28	7.03	5.04
	Ours	**2.53**	**5.41**	**4.02**
Dark	TAFNet [22]	3.97	8.10	5.80
	CCRMNet [35]	5.01	9.28	6.93
	CAGNet [25]	4.11	8.45	6.02
	MC^{3}Net [34]	3.52	7.33	5.13
	Ours	**2.54**	**5.42**	**3.84**

TABLE V
ABLATION EXPERIMENT RESULTS

Methods	GAME(0)↓	GAME(2)↓	RMSE↓
w/o MFFC	7.15	13.56	8.91
w/o CBR	4.52	8.32	5.93
w/o EFC	5.27	9.92	6.75
w/o ADFM	6.69	10.95	7.83
w/o(MFFC&EFC)	11.18	15.84	12.52
w/o(ADFM&EFC)	6.72	11.27	7.98
w/o(MFFC&ADFM)	13.91	19.76	15.72
Ours(VGGNet16)	3.87	7.05	5.22
Ours(ResNet50)	3.02	6.19	4.35
Ours(BL)	2.83	5.87	4.18
Ours	**2.42**	**5.21**	**3.54**

C. Ablation Studies

In order to validate the effectiveness of the key components in the MFCNet proposed in this article, comprehensive ablation experiments are conducted on the DroneVehicle dataset. The experimental results are shown in Table V.

1) Effectiveness of MFFC: As the key component of MFCNet, the MFFC module fully fuses multiscale feature information by interweaving attention with feature intersection. In order to demonstrate the effectiveness of MFFC, we replace it with a simple concatenation operation for low-level feature fusion. The result of the ablation experiment is shown in Table V (w/o MFFC). It can be seen that the performance of MFCNet decreases significantly after removing MFFC. In addition, to ensure the spatial consistency after feature fusion, the CBR module is designed in the MFFC module. The results of the ablation experiment (w/o CBR) show that the CBR module improves the performance of the network and is indispensable.

2) Effectiveness of EFC: In the ablation experiment, we remove the EFC from MFCNet and denote this variant as w/o EFC. As shown in the third row of Table V, after EFC is removed, there is a clear gap in all metrics, which proves the effectiveness of EFC in promoting the network to learn the deviation region between modalities. Meanwhile, the ablation result when both MFFC and EFC modules are removed simultaneously is also listed in Table V, and it is clear that the performance of the model significantly decreases and cannot meet our requirements.

3) Effectiveness of ADFM: In the ablation experiments, ADFM is replaced with MFFC and EFC, and the results are shown in Table V. It can be seen that the network performance degrades when the high-level features are underutilized. Also, the ablation results when ADFM is removed simultaneously with MFFC or EFC are shown in Table V. It is clear that the network performance is significantly poor. Only when these three modules are present at the same time, the network is able to achieve the expected high performance, fully proving the necessity of these key modules.

4) Study of Backbone Network: In this article, two parallel backbones (ConvNext) are used to extract features from RGB images and thermal images, respectively. Given that some multimodal methods use VGG16, ResNet and BL as the backbone network, we conducted ablation experiments to verify the effectiveness of ConvNext. The results in Table V show that ConvNext as the backbone outperforms other methods in all metrics, which validates its effectiveness and advancement as the backbone network for MFCNet.

5) Study of Loss Factor: In order to determine the optimal value of the loss factor α and β in loss function, we designed and conducted several experiments. These experiments aim to analyze the impact of each loss factor on the network performance so as to balance the weight of the density distribution loss L_{den} and the local similarity-aware loss L_{lc} while maintaining the prediction accuracy. As shown in Fig. 10, the height of the dots represents the performance of density estimation. It can be seen that when the α is set to 0.2 and the β is set to 0.1, MFCNet achieves the best density estimation performance. Therefore, we choose these two values as the final loss factors.

6) Study of Stage Division: We designed a staged fusion strategy to achieve optimal multimodal feature fusion. In order to determine the optimal phasing, we compare the effects of different strategies on the network performance and show the corresponding visualization results in Fig. 11, where MAAA stands for using MFFC and EFC for the first stage, and using ADFM for the final three stages, and so on. The results show that the network exhibits optimal performance when using MFFC and EFC for the first three stages and ADFM for the fourth stage, which has lower complexity than the other two strategies, so we choose this stage division strategy to build the network.

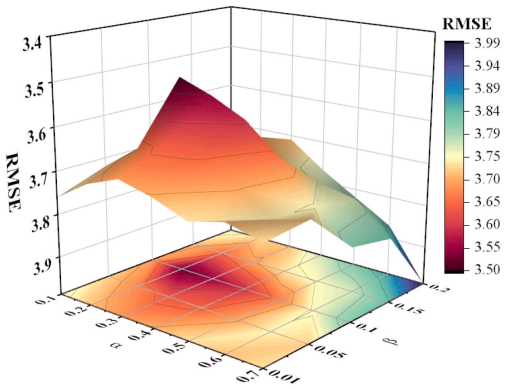


Fig. 10. Visualization of the comparison for different loss factor.

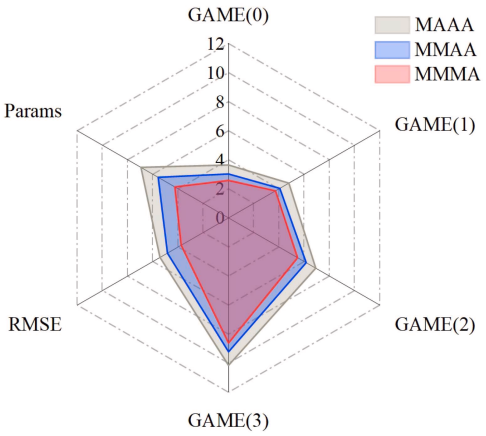


Fig. 11. Visualization of the comparison for different stage division.

D. Further Analysis of the Current Work

Although MFCNet performs well in terms of density estimation accuracy, its computational resource requirements also increase accordingly, which may be challenging in resource-limited environments. Fig. 12 visualizes the efficiency of MFCNet compared to other advanced methods, where larger bubbles represent longer GPU processing times. As is shown in the figure, MFCNet still has gaps in metrics, such as Param, FLOPs, and GPU. Therefore, in future work, we plan to further optimize MFCNet to reduce its complexity. The teacher-student learning strategy is planned to be introduced, the high-performance MFCNet is used as the teacher network, and the complex structured knowledge of MFCNet is transferred to the lightweight student network through knowledge distillation, which provides a practical solution to achieve a better balance between model accuracy and efficiency.

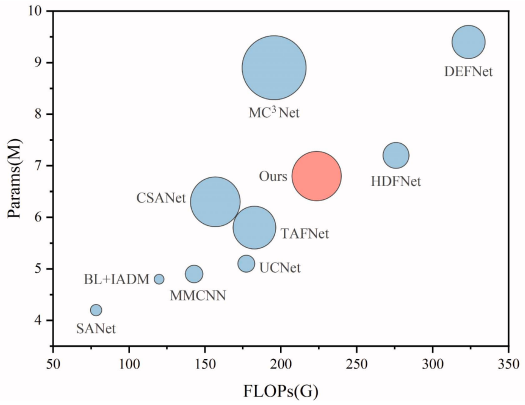


Fig. 12. Visualization of the comparison of computational resources for different networks.

V. Conclusion

In order to overcome the limitations of vehicle density estimation using only RGB images in low-light and occlusion situations, and improve the accuracy of density estimation in various scenarios, this article proposes an MFCNet for RGB-T vehicle density estimation. We choose the ConvNext network as the backbone to extract the features from RGB images and thermal images. The innovatively designed MFFC coordinates the interaction of multimodal complementary features by establishing spatial correlations, which fully integrates the multiscale information of the two modalities. On this basis, EFC aims to emphasize the attention to the vehicle misalignment region between different modalities, alleviate the spatial misalignment problem caused by fusion, and enhance the edge features. Finally, ADFM fully mines high-level features, captures deep correlations, and improves the expressiveness and robustness of the final fusion features. Experimental results on the DroneVehicle dataset show that MFCNet exhibits excellent performance compared to the existing vehicle density estimation methods.

References

[1] X. Zeng, Y. P. Wu, S. Z. Hu, R. B. Wang, and Y. D. Ye, "DSPNet: Deep scale purifier network for dense crowd counting," *Expert Syst. Appl.*, vol. 141, 2020, Art. no. 112977. [Online]. Available: https://doi.org/10.1016/j.eswa.2019.112977

[2] X. Cao, Z. Wang, Y. Zhao, F. Su, "Scale aggregation network for accurate and efficient crowd counting," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 734–750. [Online]. Available: https://doi.org/10.1007/978-3-030-01228-1_45

[3] J. Gao, Q. Wang, Y. Yuan, "SCAR: Spatial-/channel-wise attention regression networks for crowd counting," *Neurocomputing*, vol. 363, pp. 1–8, Oct. 2019. [Online]. Available: https://doi.org/10.1016/j.neucom.2019.08.018

[4] Y. Sun, B. Cao, P. Zhu, and Q. Hu, "Drone-based RGB-infrared cross-modality vehicle detection via uncertainty-aware learning," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 32, no. 10, pp. 6700–6713, Oct. 2022. [Online]. Available: https://doi.org/10.1109/tcsvt.2022.3168279

[5] K. Kwong, R. Kavaler, R. Rajagopal, P. Varaiya, "Real-time measurement of link vehicle count and travel time in a road network," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 4, pp. 814–825, Dec. 2010. [Online]. Available: https://doi.org/10.1109/tits.2010.2050881

[6] S.-L. Jeng, W.-H. Chieng, and H.-P. Lu, "Estimating speed using a side-looking single-radar vehicle detector," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 607–614, Apr. 2014. [Online]. Available: https://doi.org/10.1109/TITS.2013.2283528

[7] D. Horne, D. J. Findley, D. G. Coble, T. J. Rickabaugh, J. B. Martin, "Evaluation of radar vehicle detection at four quadrant gate rail crossings," *J. Rail Transp. Plan Manag.*, vol. 6, no. 2, pp. 149–162, 2016. [Online]. Available: https://doi.org/10.1016/j.jrtpm.2016.04.001

[8] V. Lempitsky, A. Zisserman, "Learning to count objects in images," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1–9.

[9] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 589–597. [Online]. Available: https://doi.org/10.1109/cvpr.2016.70

[10] Y. Li, X. Zhang, D. Chen, "CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1091–1100. [Online]. Available: https://doi.org/10.1109/cvpr.2018.00120

[11] N. Liu, Y. Long, C. Zou, Q. Niu, L. Pan, H. Wu, "ADCrowdNet: An attention-injective deformable convolutional network for crowd understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3225–3234. [Online]. Available: https://doi.org/10.1109/cvpr.2019.00334

[12] J. Gao, T. Han, Y. Yuan, and Q. Wang, "Domain-adaptive crowd counting via high-quality image translation and density reconstruction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 4803–4815, Aug. 2023. [Online]. Available: https://doi.org/10.1109/tnnls.2021.3124272

[13] Z. Zou et al., "Coarse to fine: Domain adaptive crowd counting via adversarial scoring network," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 2185–2194. [Online]. Available: https://doi.org/10.1145/3474085.3475377

[14] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762

[15] Y. Tian, X. Chu, and H. Wang, "CCTrans: Simplifying and improving crowd counting with transformer," 2021, *arXiv:2109.14483*.

[16] H. Lin, Z. Ma, R. Ji, Y. Wang, and X. Hong, "Boosting crowd counting via multifaceted attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19628–19637. [Online]. Available: https://doi.org/10.1109/cvpr52688.2022.01901

[17] D. Song, Y. Qiao, and A. Corbetta, "Depth driven people counting using deep region proposal network," in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, 2017, pp. 416–421. [Online]. Available: https://doi.org/10.1109/icinfa.2017.8078944

[18] D. Lian, J. Li, J. Zheng, W. Luo, and S. Gao, "Density map regression guided detection network for RGB-D crowd counting and localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1821–1830. [Online]. Available: https://doi.org/10.1109/cvpr.2019.00192

[19] H. Li, S. Zhang, and W. Kong, "RGB-D crowd counting with cross-modal cycle-attention fusion and fine-coarse supervision," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 306–316, Jan. 2023. [Online]. Available: https://doi.org/10.1109/tii.2022.3171352

[20] T. Peng, Q. Li, and P. Zhu, "RGB-T crowd counting from drone: A benchmark and MMCCN network," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 497–513. [Online]. Available: https://doi.org/10.1007/978-3-030-69544-6_30

[21] L. Liu, J. Chen, H. Wu, G. Li, C. Li, and L. Lin, "Cross-modal collaborative representation learning and a large-scale RGBT benchmark for crowd counting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4823–4833. [Online]. Available: https://doi.org/10.1109/cvpr46437.2021.00479

[22] H. Tang, Y. Wang, and L. P. Chau, "TAFNet: A three-stream adaptive fusion network for RGB-T crowd counting," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 3299–3303. [Online]. Available: https://doi.org/10.1109/iscas48785.2022.9937583

[23] W. Zhou, Y. Pan, J. Lei, L. Ye, and L. Yu, "DEFNet: Dual-branch enhanced feature fusion network for RGB-T crowd counting," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24540–24549, Dec. 2022. [Online]. Available: https://doi.org/10.1109/tits.2022.3203385

[24] H. Li, J. Zhang, W. Kong, J. Shen, Y. Shao, "CSA-Net: Cross-modal scale-aware attention-aggregated network for RGB-T crowd counting," *Expert Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119038. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.119038

[25] X. Yang, W. Zhou, W. Yan, and X. Qian, "CAGNet: Coordinated attention guidance network for RGB-T crowd counting," *Expert Syst. Appl.*, vol. 243, Jun. 2024, Art. no. 122753. https://doi.org/10.1016/j.eswa.2023.122753

[26] Z. Liu, H. Mao, C. Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11976–11986. https://doi.org/10.1109/cvpr52688.2022.01167

[27] Q. Huang et al., "Semantic segmentation with reverse attention," 2017, *arXiv:1707.06426*.

[28] B. Wang, H. Liu, D. Samaras, and M. H. Nguyen, "Distribution matching for crowd counting," in *Proc. 34th Adv Neural Inf Process Syst.*, vol. 33, 2020, pp. 1595–1607. [Online]. Available: https://doi.org/10.1109/siu53274.2021.9478013

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[30] J. Zhang et al., "UC-Net: Uncertainty inspired RGB-D saliency detection via conditional variational autoencoders," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8582–8591. [Online]. Available: https://doi.org/10.1109/cvpr42600.2020.00861

[31] D. P. Fan, Y. Zhai, A. Borji, J. Yang, and L. Shao, "BBS-Net: RGB-D salient object detection with a bifurcated backbone strategy network," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 275–292. [Online]. Available: https://doi.org/10.1007/978-3-030-58610-2_17

[32] Y. Zhang, L. Fu, Y. Li, and Y. Zhang, "HDFNet: Hierarchical dynamic fusion network for change detection in optical aerial images," *Remote Sens.*, vol. 13, no. 8, p. 1440, 2021. [Online]. Available: https://doi.org/10.3390/rs13081440

[33] Z. Ma, X. Wei, X. Hong, and Y. Gong, "Bayesian loss for crowd count estimation with point supervision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6142–6151. [Online]. Available: https://doi.org/10.1109/iccv.2019.00624

[34] W. Zhou, X. Yang, J. Lei, W. Yan, and L. Yu, "MC^{3}Net: Multimodality cross-guided compensation coordination network for RGB-T crowd counting," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 5, pp. 4156–4165, May 2024. [Online]. Available: https://doi.org/10.1109/tits.2023.3321328

[35] S. Li, Z. Hu, M. Zhao, S. Bi, and Z. Sun, "Cross-modal collaborative representation and multi-level supervision for crowd counting," *Signal, Image Video Process*, vol. 17, no. 3, pp. 601–608, 2023. [Online]. Available: https://doi.org/10.1007/s11760-022-02266-4

[36] X. Liu, J. Yang, W. Ding, T. Wang, Z. Wang, and J. Xiong, "Adaptive mixture regression network with local counting map for crowd counting," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 241–257. [Online]. Available: https://doi.org/10.1007/978-3-030-58586-0_15

[37] Q. Song et al., "To choose or to fuse? Scale selection for crowd counting," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 2576–2583. [Online]. Available: https://doi.org/10.1609/aaai.v35i3.16360

[38] J. Wan, Z. Liu, and A. B. Chan, "A generalized loss function for crowd counting and localization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1974–1983. [Online]. Available: https://doi.org/10.1109/cvpr46437.2021.00201

[39] M. Wang, H. Cai, X. Han, J. Zhou, and M. Gong, "STNet: Scale tree network with multi-level Auxiliator for crowd counting," 2022, *arXiv:2012.10189*. [Online]. Available: https://doi.org/10.48550/arXiv.2012.10189

[40] Z. Du, M. Shi, J. Deng, and S. Zafeiriou, "Redesigning multi-scale neural network for crowd counting," *IEEE Trans Image Process.*, vol. 32, pp. 3664–3678, 2023. [Online]. Available: https://doi.org/10.1109/ICIP.2017.8296324

[41] Z. Xie et al., "BGDFNet: Bidirectional gated and dynamic fusion network for RGB-T crowd counting in smart city system," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–16, Jun. 2024. [Online]. Available: https://doi.org/10.1109/TIM.2024.3418111

[42] Y. Pan, W. Zhou, M. Fang, and F. Qiang, "Graph enhancement and transformer aggregation network for RGB-thermal crowd counting," *IEEE Geosci. Remote Sens. Lett.*, vol. 21, Feb. 2024, Art. no. 3000705. [Online]. Available: https://doi.org/10.1109/LGRS.2024.3362820

Ling-Xiao Qin was born in Shandong, China, in 1998. She received the B.S. degree from Shandong University of Science and Technology, Qingdao, China, in 2022, where she is currently pursuing the M.S. degree.

Her research interests include image processing and deep learning.

Hong-Mei Sun received the B.S. and M.S. degrees in computer science from Shandong University of Science and Technology, Qingdao, China, in 1995 and 2005, respectively.

She is currently an Associate Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology and the Leader of the Key Research and Development Projects, Shandong Province, China. She has more than 20 first-author publications and 50 co-author publications. Her research interests include computer vision, deep learning, and software engineering.

Xiao-Meng Duan was born in Shandong, China, in 1999. She received the B.S. degree from Shandong University of Science and Technology, Qingdao, China, in 2022, where she is currently pursuing the M.S. degree.

Her research interests include image processing and deep learning.

Cheng-Yue Che was born in Shandong, China, in 2001. She received the B.S. degree from Shandong University of Science and Technology, Qingdao, China, in 2023, where she is currently pursuing the M.S. degree.

Her research interests include image processing and deep learning.

Rui-Sheng Jia received the B.S., M.S., and Ph.D. degrees in computer science from Shandong University of Science and Technology, Qingdao, China, in 1995, 2002, and 2010, respectively.

He is currently a Full professor with the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China, and the Leader of the Natural Science Foundation, Shandong Province, China. He has more than 30 first-author publications and 50 co-author publications. His research interests include artificial intelligence, computer vision, information fusion, and microseismic monitoring and inversion.

A Trusted OS Penetration Testing Scheme Based on Metasploit and BeEF

1st Hengji Miao
School of Cyberspace Security,
Shandong University of Political Science and Law
Jinan, China
miaohengji@qq.com

2nd Lei Shang*
School of Cyberspace Security,
Shandong University of Political Science and Law
Jinan, China
leilishang@163.com

3rd Weihong Gan
School of Computer Science,
Qufu Normal University
Rizhao, China
ganweihong0330@163.com

4th Chenle Zhang
School of Cyberspace Security,
Shandong University of Political Science and Law
Jinan, China
zhangchenle@foxmail.com

5th Zhuo Guan
School of Cyberspace Security,
Shandong University of Political Science and Law
Jinan, China
guanzhuosz@qq.com

6th Zhihong Ge
School of Cyberspace Security,
Shandong University of Political Science and Law
Jinan, China
gezhihongxx@foxmail.com

***Abstract*—Hacker attacks have become more frequent in recent years, and network intrusions caused by vulnerabilities in the operating system have increased dramatically. Most of the protection currently applied to the Windows operating system is to download virus detection software from security vendors. However, most of this type of software detects the files downloaded by the user and has little effect on preventing attacks initiated by hackers utilizing operating system vulnerabilities. Therefore, this paper proposes a detection scheme based on BeEF and Metasploit to help users easily and accurately detect possible system vulnerabilities on the operating system, and to help users cut off the possibility of hackers utilizing system vulnerabilities to launch attacks at the source with low operational complexity.**

Keywords-cybersecurity; penetration testing; operating systems; Metasploit; BeEF

I. INTRODUCTION

The operating system serves as the basis for the use of electronic devices by individual users, and its security is susceptible to viruses, Trojan horses, network intrusions, and illegal human operations. Once an operating system is misconfigured or has an unpatched or undiscovered vulnerability, it can lead to the collapse of the entire security system. Therefore, it is urgent and necessary to be able to discover the security vulnerabilities of the operating system and judge its security status in a timely and accurate manner.

Currently, most of the solutions applied to operating systems by individual users are to use security software developed by security vendors or to hire professionals from the vendors to perform security services. Many researchers at home and abroad have also conducted a lot of research on operating system security. Yijun He utilized the network security tools that come with Kali Linux to implement vulnerability attacks on mainstream operating systems (Windows Sever2003, Windows7 SP1, Metasploitable2-Linux) and conduct penetration tests[1]. Yanying Ma[2], Cong Wang[3], et al. carried out secondary development of Metasploit, assembled current popular security tools, and designed a network security assessment system. Xiaolan Li, Jingcheng Fang, and Xiong Cai et al. verified the embedded operating system at the unit level, module level, and system level, respectively, using VCC, CBMC, and PAT tools[4].

However, there are still problems with these solutions; Alhamed et al. showed that network penetration testing tools may have performance bottlenecks when dealing with complex network topologies, resulting in the inability to comprehensively detect all vulnerabilities[5]; and Eshghie et al. pointed out that dynamic vulnerability detection tools may miss these potential security hazards when confronted with dynamically generated code or untriggered code paths[6]; Huang et al.'s study, on the other hand, shows that existing fuzz-testing tools have a high rate of false positives when dealing with multiple protocols or mixed data streams, and may misclassify legitimate network traffic as malicious behavior[7]. The above solutions more or less suffer from the problems raised by these researchers, while the highly specialized nature of the tools may also make it difficult for individual users to operate them.

Therefore, this paper proposes a trusted OS penetration testing scheme based on Metasploit and BeEF, which utilizes the linkage of these two tools in the Kali Linux operating system to achieve reliable penetration testing of mainstream operating systems. Linking the functions of Metasploit with BeEF, so that the vulnerability modules in Metasploit are displayed in BeEF, users can dynamically view what vulnerabilities may exist in the target host operating system, and combine with the vulnerability modules that come with BeEF to form a set of visualized, highly reliable, and low-professionalization of the operating system penetration testing solution. Makes it simple and easy for individual users to view the security vulnerabilities that exist on their operating systems.

II. RELATED TECHNOLOGIES

A. Penetration Testing Techniques

Penetration testing is an authorized simulated attack on a computer system designed to evaluate the security of the target host.

979-8-3315-2832-4/24/$31.00 ©2024 IEEE
DOI 10.1109/ICBCTIS64495.2024.00050

Penetration testing can be categorized into black-box testing, white-box testing, and stealth testing, where black-box testing means that the penetrator is in a state of ignorance about the system. White-box testing is the exact opposite of black-box testing, where the white-box test is allowed to obtain various information about the target through normal channels. Stealth testing is conducted without informing the majority of the target organization's employees and is therefore an effective way to check whether the target organization can monitor, respond, and recover from an information security incident.

B. Trusted Operating System

Operating system trustworthiness is the ability of the system's software and hardware to function in the manner intended by the original design.

The foundation of trustworthiness is that the software integrity of the system is good and has not been intruded on or tampered with. In this paper, we focus on software trustworthiness, i.e., whether the operating system can run smoothly according to its original set of security protection strategies.

C. Metasploit Technology Framework

Metasploit (MSF) is an open-source security vulnerability detection tool that comes with hundreds of known software vulnerabilities. Through Metasploit, people can easily implement attacks on computer software vulnerabilities. Metasploit is written in Ruby language, has good cross-platform, and can be installed in a variety of operating systems to use. Widely used for penetration testing, vulnerability research, and red team exercises, Metasploit enables users to effectively assess and strengthen the security of their systems thanks to its powerful automation and scripting capabilities. Metasploit's extensive community support and continuous updates make it a critical tool in the information security space, helping users prevent potential security threats before actual attacks occur.[8]

Metasploit framework consists of a base library file, modules, plug-ins, interfaces, functional programs composed of five parts, and other security tools through the plug-in form of interconnection with it, as shown in Fig. 1.

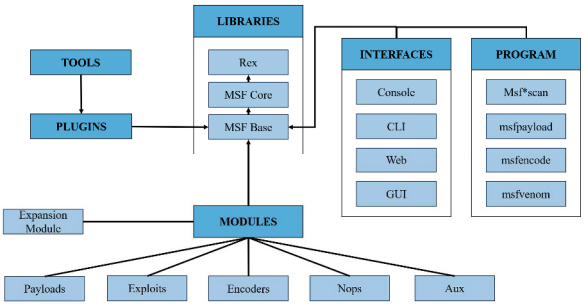


Figure 1. Metasploit Framework.

D. BeEF Technical Framework

BeEF, whose full name is The Browser Exploitation Framework, is a penetration testing tool for browsers that offers several visualization modules. It allows security researchers and penetration testers to perform in-depth client-side security assessments by taking control of the victim's browser.BeEF can integrate with other security tools to help users identify and fix vulnerabilities in browsers and their plug-ins, improving overall cyber security protection.[8]

At the same time, BeEF supports API calls, enabling users to write their modules. BeEF utilizes the XSS technique to establish contact with the target host, the process is shown in Fig. 2.

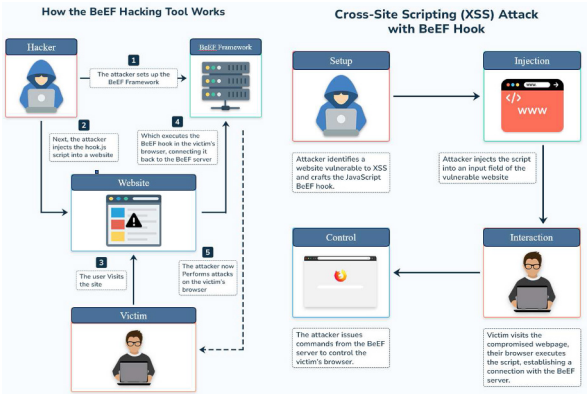


Figure 2. BeEF Attack Flow.

The connection is established when the target host accesses the BeEF's hook file. Thereafter, the target host will be displayed in the BeEF's management interface when it comes online. Internally, BeEF can detect which command modules work in the currently victimized browser and indicate them with different colors, as shown in Table Ⅰ.

TABLE I. COLOR MEANING

Color	Hidden Meaning
Green	The command module can be run on the target browser without any exception for the user.
Orange	The command module will work on the target browser, but the user may feel anomalies (e.g. there may be pop-ups, prompts, jumps, etc.).
Gray	The command module has not been validated for this target, i.e., it is not known if it can be run.
Red	This command module does not apply to this target.

III. OPERATING SYSTEM PENETRATION TESTING EXPERIMENT

This section includes three experiments, which are Metasploit remote control of Windows systems, BeEF using reflective XSS to take privileges, and BeEF linking Metasploit for vulnerability discovery.

The experimental environment and tools are shown in Table Ⅱ and Table Ⅲ.

TABLE II. EXPERIMENTAL ENVIRONMENT

Test Environment	IP
Hosts: Kali-Linux-2024.1-installer-amd64	192.168.244.133
Target Drone: Windows 10 pro 22h2	192.168.244.135
Target Drone: Metasploit-linux-2.0.0	192.168.244.136
Target Drone: Windows XP Home Edition	192.168.244.140

TABLE III. EXPERIMENTAL TOOLS

Tool	Releases
Metasploit	v6.3.55-dev
BeEF	0.5.4.0
VMware Workstation	17 Pro 17.5.0 build-22583795

A. Metasploit Remote Control for Windows

First, use the "msfvenom" command within Kali Linux to generate the controlled terminal. To move the generated payload to Windows 10, connect the steps as follows: Use the handler in the multi-directory under exploit as the active end; Set the payload to the name of the reverse remote control program you just used; Set up lhost and lport; Open the listening, as shown in Fig. 3.

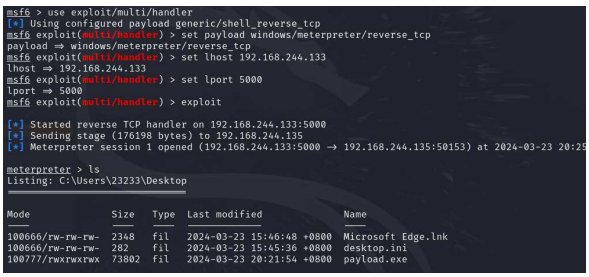


Figure 3. Start the Master and Execute the Remote Control Software.

B. BeEF Utilizes Reflective XSS to Gain Privileges

This experiment will demonstrate the use of the XSS-reflected vulnerability in DVWA, using the DVWA integrated with Metasploit-linux-2.0.0, with the difficulty set to medium. Start BeEF in Kali Linux in the location of the input box to fill in the BeEF hook address and submit, after the submission of the interface.

Back to the BeEF side, you can see that the privileges of the virtual machine with DVWA have been obtained, select the Get Cookie command and execute it, you can find that the cookie information is returned correctly, indicating that the privileges have been obtained successfully, as shown in Fig. 4.

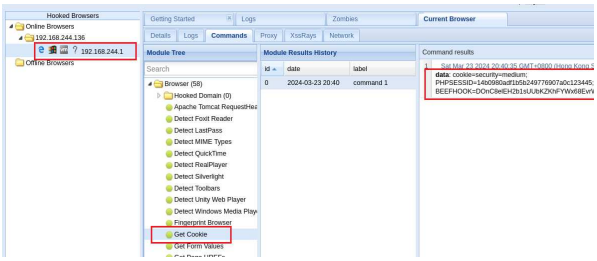


Figure 4. Get Cookie.

C. BeEF Utilizes Reflective XSS to Gain Privileges

Firstly, Link BeEF with Metasploit, change BeEF's configuration file, turn on the load module of MSF in the extension, and change the host in http to Kali's IP address. Secondly, Modify the MSF configuration file in the BeEF extension file, change enable to true, ssl to false, and add the path for custom to make it link to the Metasploit module. Then, Load the module in Metasploit. After loading is complete jump to the /usr/share/beef-xss folder and start BeEF via "./beef".

Open the Metasploit section of BeEF to see the newly added vulnerability module, and then simply select the vulnerability and fill in the corresponding parameters to exploit the vulnerability. The user simply establishes a connection between the host to be detected and BeEF, determines whether the vulnerability is contained on the target host by the color of the orb as described above, and then decides on the necessity of fixing this vulnerability based on the hazards of running it.

Take MS12-063 as an example, select MS12-063 Microsoft Internet Explorer exec-Command Use-After-Free Vulnerability in BeEF, and fill in the corresponding parameters on the right side, as in Fig. 5.

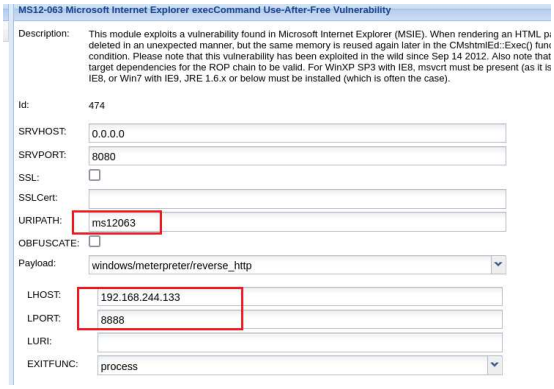


Figure 5. MS12-063.

This action will generate a malicious attack page: http://192.168.244.134:8889/ms12063, which will be accessed remotely when the target machine accesses the URL. Accessing the URL in Windows 10 and returning to Kali Linux after a successful visit reveals that no session was fetched in Metasploit's terminal, indicating that the vulnerability does not exist in Windows 10. Visiting the URL in Windows XP, the session can be fetched in the Metasploit terminal, indicating that the vulnerability exists in Windows XP. After connecting Windows XP to the BeEF, you can also see that the dot corresponding to MS12-063 is green, which means that it exists in this target host.

Therefore, the process of operating system penetration testing by BeEF linked to Metasploit can be summarized as shown in Fig. 6.

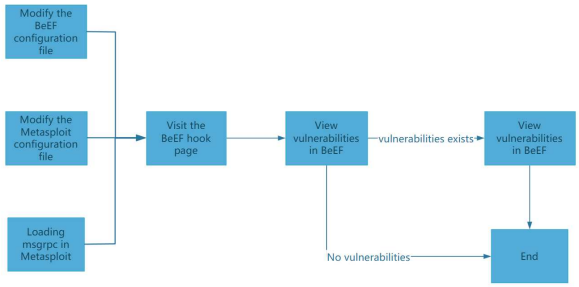


Figure 6. BeEF and Metasploit Linked Penetration Testing Process.

IV. ANALYSIS OF RESULT

Comprehensive experiments can be seen, Metasploit has a perfect vulnerability library, and penetration testing only needs to select the exploit, and set the corresponding parameters can be. However, when performing penetration testing of operating systems, it is less efficient to try each possible vulnerability of the target system.

When using BeEF for penetration testing, its friendly color-coded design makes it easy to view the exploitable vulnerabilities on the website, but it is difficult to detect the vulnerabilities of certain services in the operating system.

By combining the two, the vulnerability can be determined by simply connecting the target host to BeEF and by the color of the dot in front of each function. Grayed-out vulnerabilities can be manually verified by simply configuring parameters, greatly reducing the time it takes to find vulnerabilities.

Table Ⅳ compares Metasploit and BeEF linkage with other commonly used network security tools around host scanning, password cracking, web scanning, and social engineering. It can be seen that the program not only has advantages in the direction of the research in this paper but will also be a comprehensive attack tool in its own right. When conducting Red-Blue confrontation and net protection operations, members of the Red side can utilize this combination of tools to attack the Blue side.

TABLE IV. EXPERIMENTAL TOOLS

Table Head	Table Column Head								
	Host Scanning	*Password Cracking*	*Web Scan*	*Social Engineering*	*Vulnerability Discovery*	*Exploit*	*Session Control*	*Report Generation*	*Visualization Interface*
Nmap	√	×	×	×	×	×	×	√	×
Zenmap, X-Scan	√	×	×	×	×	×	×	√	√
Hydra	×	√	×	×	×	×	×	×	×
xHydra, Cain	×	√	×	×	×	×	×	×	√
w3af console	×	×	√	×	√	√	×	√	×
WebInspect, Safe3SI	×	×	√	×	√	√	×	√	√
SET	×	×	×	√	×	×	×	×	×
NESSUS	√	×	√	×	√	×	×	√	√
Metasploit	√	√	×	×	√	√	√	√	√
BeEF	×	×	√	√	√	√	√	×	√
BeEF & Metasploit	√	√	√	√	√	√	√	×	√

According to the data in Table Ⅳ, the weighted sum formula is used to calculate the weighted sum of each system, and the attributes in Table 4 are represented by "H, P, W, S, V, E, C, R, G" from left to right, the value of 1 indicates that it has this function, and the value of 0 indicates that it does not have this function, and the corresponding weights are as follows

$$w_H = 0.20, w_P = 0.10, w_W = 0.20, w_S = 0.10, w_V = 0.20, w_E = 0.10, w_C = 0.05, w_R = 0.10, w_G = 0.05 \quad (1)$$

and satisfy

$$w_H + w_P + w_W + w_S + w_V + w_E + w_C + w_R + w_G = 1 \quad (2)$$

The above weighting assignments are based on industry standards, common security threats and their impact, and the contribution of the function to overall system security. This allocation ensures that penetration testing focuses on the most critical security areas, while also covering other important supporting functions to provide a comprehensive security assessment.

The operational complexity level O is calculated by the formula:

$$O = w_H \cdot H + w_P \cdot P + w_W \cdot W + w_S \cdot S + w_V \cdot V + w_E \cdot E + w_C \cdot C + w_R \cdot R + w_G \cdot G \quad (3)$$

The larger the value of O, the more suitable the tool is for OS penetration testing. Using the above formula, the calculation results are obtained as shown in Fig. 7.

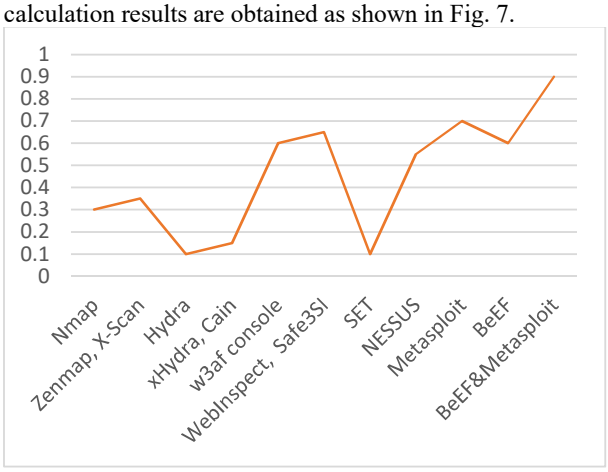


Figure 7. O Calculation Results.

According to Fig. 7, it can be seen that BeEF and Metasploit used in conjunction will have a better performance in OS penetration testing.

In summary, the penetration scheme is functionally perfect in covering all possible attack surfaces. Also in terms of operability, BeEF's graphical interface allows users to conveniently view possible vulnerabilities. Therefore,

using BeEF Linkage Metasploit to verify the trustworthiness of the operating system software level is both reliable and convenient, and its user-friendly visualization interface will make the test results clear at a glance. However, its degree of automation is low, requiring users to have a certain degree of professional knowledge, pending more in-depth optimization and research in the future.

V. SUMMARY

In today's world of increasing cyber-attacks and increasingly severe cyber-security situations, the trustworthiness of the operating system is an important line of defense against attacks. In this paper, we focus on penetration testing of operating systems using Metasploit in conjunction with BeEF to detect whether they are trustworthy at the software level. For network security staffing is not a well-staffed organization, using the methods described in this article, the general security personnel, and testers can complete the initial detection, to help enterprises reduce the possibility of intrusion.

However, there are still some flaws in this program, which can cause some potential problems if Metasploit's vulnerability repository is not updated on time. Therefore how to effectively synchronize the latest Metasploit libraries is a future research priority. In addition, this method still requires personnel with a certain level of safety knowledge to operate, so minimizing the threshold of operation and increasing the level of automation is also a priority for future improvement.

ACKNOWLEDGMENT

This research was financially supported by the Scientific Research Project of Shan Dong University of Political Science and Law(2019Z01B) and the Teaching Reform Project of Shandong University of Political Science and Law(2018JGA002).

REFERENCES

[1] Y. He, "The Research on penetration testing based on Kali Linux," M.S. thesis, Central South Univ. of Forestry & Technol., Changsha, China, 2019. (references)

[2] Y. Ma, "The Design and Implementation of Network Security Assessment System Based on Metasploit," M.S. thesis, Hebei Univ. of Sci. and Technol., Shijiazhuang, China, 2014.

[3] C. Wang, "Design and Realization of a Penetration Test Platform Based on Metasploit," M.S. thesis, Univ. of Chinese Acad. of Sci., Beijing, China, 2018.

[4] Y. Wang, J. Fang, X. Cai, Z. Zhang, Y. Cai, and W. Miao, " A formal verification method for embedded operating systems," Journal of East China Normal Univ. (Natural Science), vol. 11, Dec. 2023, pp. 1-17.

[5] M. Alhamed and M. M. H. Rahman, "A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions," Applied Sciences, vol. 13, no. 12, Jun. 2023, pp. 6986-7009, doi:10.3390/app13126986.

[6] M. Eshghie, C. Artho, and D. Gurov, "Dynamic vulnerability detection on smart contracts using machine learning," in Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering, Jun. 2021, pp. 305-312, doi:10.1145/3463274.3463348.

[7] Y. Huang, B. Jiang, and W. K. Chan, "Eosfuzzer: Fuzzing EOSIO smart contracts for vulnerability detection," in Proceedings of the 12th Asia-Pacific Symposium on Internetware, Nov. 2020, pp. 99-109, doi: 10.1145/3457913.3457920.

[8] M. Buckbee, Feb. 2022, "What is Metasploit? The Beginner's Guide," https://www.varonis.com/blog/what-is-metasploit.

[9] R. Dezso, May. 2024, "How to Use the BeEF Hacking Tool," https://www.stationx.net/beef-hacking-tool.

A Blockchain-based Electronic Data Forensics System Design and Implementation

Lei Shang, Xiaoyan Yang, Xuanrong Chen
School of Cyberspace Security
Shandong University of Political Science and Law
Jinan, China
leilishang@163.com, xyyang@sdupsl.edu.cn, 275553270@qq.com

2023 International Conference on Data Security and Privacy Protection (DSPP) | 979-8-3503-0315-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/DSPP58763.2023.10405059

***Abstract*—With the rapid development of the digital economy, in judicial practice, the type of evidence is developing from physical evidence to electronic evidence. Compared with traditional physical evidence, electronic evidence is vulnerable to external network attacks and tampering by internal practitioners in the process of collection, fixation, storage, transmission, etc. Therefore, determining the authenticity of electronic evidence is costly and difficult to admit, which directly affects the proportion of electronic evidence admissible in litigation. The unique characteristics of blockchain technology, such as tamper-proof, traceability, and multi-party participation, naturally fit with the demand for electronic data deposition. The electronic data deposit based on blockchain technology can avoid evidence forgery and reduce the impact of network attacks, ensure the authenticity of electronic evidence, and improve the admissibility rate of electronic evidence. This paper proposes a blockchain smart contract-based deposition system, which can realize the digital fingerprint deposition of documents by users, and can also display the timestamp of the evidence while the evidence is on the chain as well as the forensic evidence, thus ensuring the traceability and security of the evidence on the chain. Improve the efficiency and security of electronic data deposition.**

Keywords—Blockchain, Electronic Evidence, Smart Contracts, Security, Forensics

I. INTRODUCTION

Entering the era of information technology and digitalization, more and more evidence is presented as electronic evidence in judicial practice, and its specific manifestations are increasingly diversified, with a significant increase in the frequency of use and data volume. Electronic evidence has virtual characteristics compared to traditional evidence, and its authenticity is often more easily questioned. Electronic evidence due to digital features can be quickly disseminated, accurately reproduced, easily damaged, and easy to lose, its authenticity, integrity is difficult to confirm. With the development of science and technology, electronic evidence of new patterns also continues to emerge, manifested in massive data-based, machine algorithms as tools, etc., which brings a large number of access to the demand for evidence. In judicial practice, on the one hand, the judge for the authenticity of electronic data takes a very strict attitude, the admissibility rate of electronic evidence is low. On the other hand, to ensure the security of electronic evidence, the procedure of electronic data review is too much, and the admission process when accessing evidence is quite cumbersome and inefficient.

Blockchain is a decentralized distributed ledger technology. Compared with traditional centralized databases, blockchain is decentralized, tamper-proof, traceable, multi-party maintenance, and open and transparent through the application of traditional technologies such as distributed data storage, P2P transmission, consensus mechanisms, encryption algorithms, and smart contracts[1]. In essence, both blockchain and law are trust mechanisms, and using blockchain technology as the storage method of electronic evidence can ensure the integrity, authenticity, and suitability of evidence. Electronic evidence on the chain can prevent human tampering and can be traced back to the root cause and be verified at any time. Through the consensus mechanism, each credible institution and judicial authority on the chain can save the time spent on verifying the evidence and enhance the legal validity and judicial efficiency of the electronic evidence.

In recent years, blockchain technology has received a lot of attention around the world. Based on blockchain, Jian Wang et al. proposed a secure storage system for electronic data applicable to judicial trial scenarios, including a judicial alliance chain called JudChain and an evidence storage chain named EviChain, which can realize efficient management and secure storage of judicial process data and electronic evidence. [2]. Jiongshu Chen designed a decentralized identity management module for maintaining the identity information of networked entities, thus ensuring the secure transmission of data by entities in the network, and enabling electronic evidence to be used directly as valid evidence in the final court review process[3]. Kunqiao Yang designed a blockchain technology-based electronic evidence deposition system that uses blockchain technology and RBAC authority control to guarantee secure, trustworthy, and traceable data [4,5]. There is also some literature that aims to elaborate on blockchain forensics in different application areas, Lamprini Zarpala provides an Ethereum-based forensic system for the financial crime domain with an integrated standardized forensic process and chain of custody preservation mechanism [6]. Ronghong Xu proposed a V2G (Vehicles to Grid, V2G) forensic-oriented blockchain security authentication scheme for the deposition of transaction records in the energy trading market [7].

For the current dilemma of judicial deposition, the biggest advantage of blockchain technology is to give electronic evidence a "fingerprint" and to ensure that the data written in the blockchain will not be changed, which is mainly achieved through hash value and decentralization. This paper proposes a blockchain smart contract-based deposition system, which can realize the active deposition of digital fingerprints of documents by users and is suitable for forensic, audit, notary, arbitration, and other institutions or judicial organs to obtain and verify data. The method proposed in the paper is experimentally tested for functionality, and the trustworthiness and security of the system are verified.

II. RELATED TECHNOLOGIES

A. Blockchain

Blockchain is an innovative application model of computer technology in the Internet era, combining data encryption, distributed storage, peer-to-peer transmission,

Funded Projects: Scientific Research Project of Shan Dong University of Political Science and Law(2019Z01B), Teaching Reform Project of Shandong University of Political Science and Law(2018JGA002), Shandong Provincial Natural Science Foundation Major Basic Research (ZR2019ZD10).

consensus mechanism, and other technologies. Blockchain has a very broad development prospect in academic research and practical application and is considered an important cornerstone and a powerful driver for the transformation of information Internet to value Internet. At present, blockchain is used in industrial Internet, traceability and anti-counterfeiting, intelligent transportation, healthcare, agriculture, and other fields. Although the architecture of blockchain in different application scenarios is not the same, there are still many commonalities. The blockchain platform is usually divided into 5 layers, which are the data layer, network layer, consensus layer, contract layer, and application layer [1].

B. Ethereum

Ethereum is a universal blockchain platform with a collection of smart contract functions and Turing-complete smart contract execution capabilities, allowing decentralized applications to be built on the Ethereum platform through smart contracts. The Ethereum Virtual Machine (EVM) provides a secure sandbox environment for executing smart contract code, and each transaction or smart contract execution is executed in the EVM [1].

C. Smart Contracts

Smart contracts are a key element of Ethernet, providing programmability to the blockchain and expanding the application scenarios of the blockchain. Smart contracts use conditional programming (if y, then x) to perform operations that need to be executed after a predefined set of parameters is met. Blockchain-based smart contracts are invariant, autonomous, and transparent, with more emphasis on transactions. The input, output, and state of the contract are stored in the blockchain and rely on consensus algorithms between nodes to complete[8,9].

D. Timestamp Service

A timestamp is complete and verifiable data that can indicate that electronic data has existed at a specific point in time. The trustworthy timestamp can determine the precise time of electronic document generation and prevent the electronic document from being tampered with, providing credible time proof and proof of content authenticity and integrity for electronic data. The use of electronic evidence is authoritative and trustworthy, in line with the requirements of the Electronic Signature Law, and legally validated.

Timestamps not only can accurately mark the occurrence time of the act but also can build a credible and complete evidence chain through the sequence of time, which is an important technical service for electronic data deposition.

E. web3.js

The emergence of Ethereum redefined web3 as the underlying network and value network for blockchain-based value delivery. web3.js is a JSON-RPC wrapper based on JavaScript and Node.js. web3.js enables user interaction with the block link port, and JSON-RPC enables connected communication with smart contracts. JSON-RPC is a stateless, lightweight remote procedure call protocol (RPC) that is transport protocol independent and can use sockets, HTTP, or other protocols, and it uses JSON (RFC4627) as the data format. As shown in Fig.1.

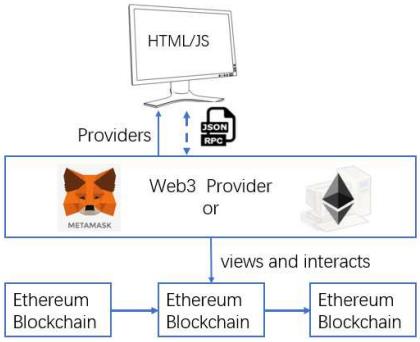


Fig. 1. web3.js Principle.

III. BLOCKCHAIN-BASED ACCESS CERTIFICATE MODEL

A. Model Construction

In the traditional judicial procedures involving electronic evidence, before the electronic data is applied to the litigation process, it is generally necessary to go through a number of links, from the collection of the original electronic data, the user or the third-party authentication agency fixed preservation, to the public security organs, the procuratorate transmission, and then to the court's access to the cumbersome procedures and the flow of both increased the risk of failure of the electronic evidence has also led to the inefficiency of the judiciary [2,10]. In particular, the authenticity appraisal of electronic evidence is a professional work, which is usually done by public security organs or commissioned professional organizations, and provides appraisal documents. Regarding the authenticity of the flow process, it is mainly guaranteed by the rules and measures formulated in the judicial process, such as minimizing the flow and making good records in the flow process. Constructing a blockchain-based forensic service system, as shown in Figure 2, can collect and keep electronic evidence more effectively and safely, and provide a guarantee for the authenticity of electronic evidence carriers.

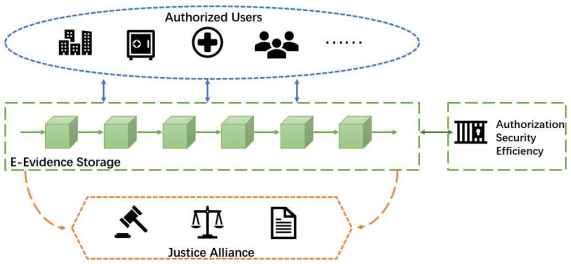


Fig. 2. Blockchain-based access certificate service system.

Real-name users can store electronic evidence and query on-chain evidence through the electronic evidence storage system with the support of blockchain technology for use in future transaction disputes or judicial proceedings. Taking copyright protection as an example, creators can upload information about their works on the chain and automatically generate digital fingerprints and time stamps. In case of copyright disputes, the on-chain evidence can be used as proof for defending rights. The depository completes user verification, audits the uploaded data, responds to users' storage query requests and provides authorized use, safe and efficient access to the evidence service. Judicial institutions access the service interface to realize the process of

depositing, taking, showing, and questioning electronic evidence in judicial practice, which corresponds to the workflow of storing, extracting, presenting, and questioning electronic data in the system. At the same time, uploading the transaction and cooperation data between institutions forms traceability of the behavior in the judicial process, which not only improves judicial efficiency but also strengthens the credibility of the judicial process with the support of blockchain technology.

B. Working Process

The system designed in this paper will run on the Ether private chain, as shown in Fig.3. The smart contract written in solidity language (an object-oriented programming language) is tamper-evident in the contract layer of the blockchain system, and in the Byzantine fault-tolerant consensus mechanism, only when the user controls more than half of the nodes can the electronic evidence stored in the nodes be changed, which can effectively guarantee the authenticity of the data.

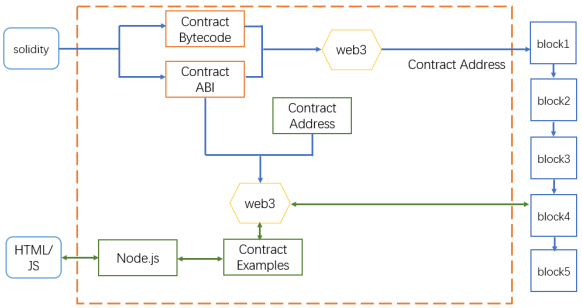


Fig. 3. Working Process.

In terms of working modes, one is the "independent deposit mode", whereby parties can access the blockchain in their own names and deposit evidence on their own, which is conducive to the parties' more proactive participation in judicial activities, and further prevents the tampering or deletion of electronic evidence as a result of judicial corruption. The other is the "third-party deposit model", which means that litigants cannot directly upload evidence onto the blockchain, but need to purchase deposit services from the data service provider designated by the court, and the data service provider will deposit the evidence on behalf of the litigants. The parties will obtain a "certificate of deposit" as proof of the authenticity of the evidence. The first model, focuses on the convenience of the deposit, and the second model emphasizes strict management, but no matter which working mode, the core is the interaction with the smart contract with the deposit function.

IV. EXPERIMENTAL DATA AND ANALYSIS OF RESULTS

A. Experimental environment

The experiments in this paper are based on the Node.js truffle framework to build decentralized applications, and the web front-end application interface is implemented through JSON-RPC requests to deploy and invoke smart contracts on Ethernet nodes, and the specific hardware and software environment configurations are shown in Table I.

TABLE I. HARDWARE AND SOFTWARE ENVIRONMENT FOR EXPERIMENTS

Category	*Name*	*Model/version*	*Function*
Hardware	CPU	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz	Hardware support for project operation
	RAM	8.00 GB	
	OS	Windows10 64bit	
software	Node.js	v18.15.0	Provide development environment
	Ganache.exe	V2.5.4	Provide Taiwans private chain
	truffle	v5.8.1	Ethernet Development Framework
	Vs code	v5.8.1	Lightweight editor with solidity language support
	Nginx	v1.22.1	A lightweight web server

Ganache is a simulator of the Ethernet runtime environment, which simulates the behavior and interface of the Ethernet public chain. Smart contracts and user interactions can be easily debugged by linking and interacting with this simulated environment, and verifying the connection results are shown in Fig.4 below, with 10 accounts automatically created by default.

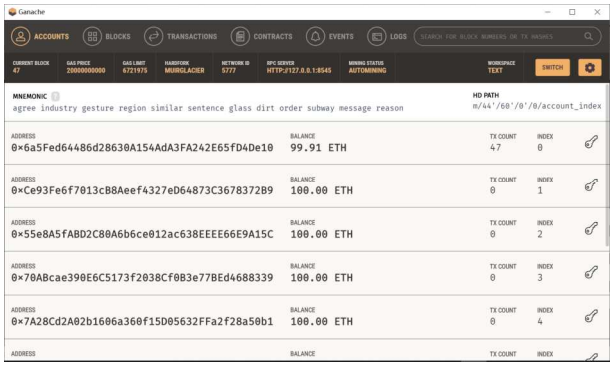


Fig. 4. Ethernet Browser and Account Information.

B. Main code analysis

1) Smart contract deposition function and forensic function

A state variable of string type is used to declare the extracted data fingerprint, store the value of the variable when executed, and return the generated transaction hash to enable the data fingerprint to be deposited on the Ethernet private chain. The forensic function is used to obtain the values of the variables in the deposit function, returning the data fingerprint and transaction timestamp stored in the block. As shown in Fig. 5

```
function addDocHash (bytes32 hash) public {
    Record memory newRecord = Record(block.timestamp,
block.number);
    docHashes[hash] = newRecord;
  }

  function findDocHash (bytes32 hash) public view
returns(uint, uint) {
    return (docHashes[hash].mineTime,
docHashes[hash].blockNumber);
```

Fig. 5. Deposition function and forensic function.

2) Define the web3 provider

Setting up Web3's Provider in Web3.js is used to tell our code which node to interact with. Getting a contract instance

requires the contract address and the contract ABI (Application Binary Interface). The function of the ABI is to tell Web3.js how to format function calls in a way that the contract understands, and to get data from the contract. As shown in Fig. 6.

```
function init () {
  //set up network
  let provider = new
Web3.providers.HttpProvider("http://localhost:8545");
  web3 = new Web3(provider);

  //contract abi
  let abi = [
.....
```

Fig. 6. web3 provider.

C. *Analysis of Experimental Results*

1) Contract Writing and Deployment

First, the contract is written and compiled. Deploy the contract to the blockchain. After successfully deploying the contract, it returns information such as account information ("0x 6a5Fed64486d28630A154AdA3FA242E65fD4De10"), contract address ("0xDe91d0Ba5D330E3c375Df5b3626FF21c32f2D7e8"), block number ("32"), etc. Experiment The results show that the command window is consistent with the blockchain browser. As shown in Fig. 7.

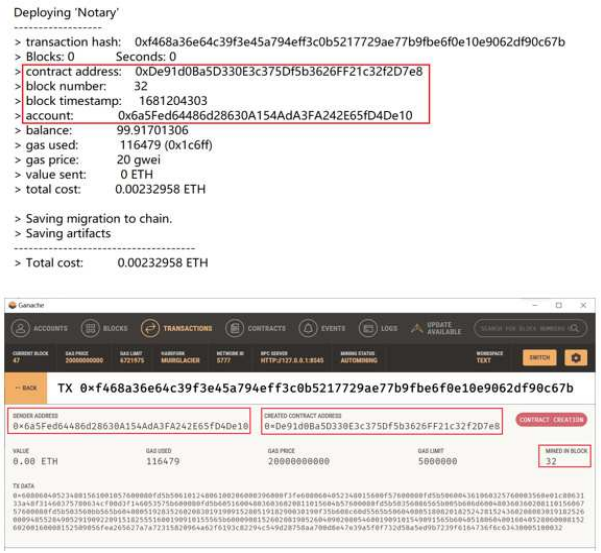


Fig. 7. Contract Deployment.

2) Select depository documents

A front-end interactive platform for experiments is built to allow functional testing of the system's smart contract layer. The front-end page can select the electronic documents that need to be deposited, verify the upload and deposition results of electronic evidence, and display forensic results. As shown in Fig. 8.

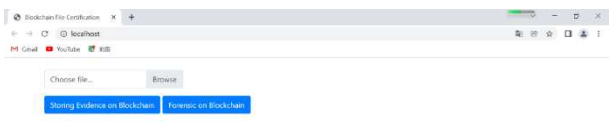


Fig. 8. web page.

3) Electronic Evidence Storage

To extract digital fingerprints from uploaded electronic evidence. Click "Storing Evidence on Blockchain" and the system returns the file's digital fingerprint File hash value. "6674fcb2f5f3828edd19b3b077947592cc6be89f19cdc28d55e6e4f0634d484a", Transaction ID:" 0x02e3216cc79528cd66286805b1ee5701e90ba4c6c4a6379ced8fe9ec64b2fe21". After filehash.exe offline verification, the file digital fingerprint extraction results are accurate. As shown in Fig. 9.

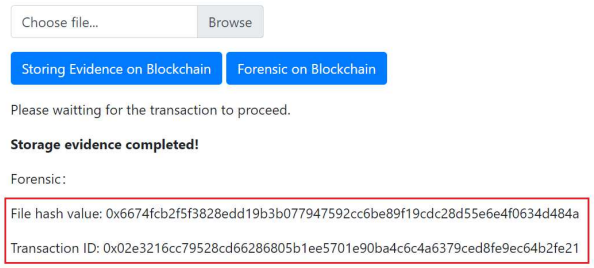


Fig. 9. Results of deposition.

4) Electronic Evidence Forensics

Click "Forensic on Blockchain" to call the forensic function to get the latest data fingerprint of the block. The return value is the input of the forensic function and does not involve the change of the contract state value, this contract call is not a transaction, no gas is consumed, and the execution result of forensic returns Block number and time stamp. As shown in Fig. 10.

Fig. 10. Results of Forensic.

5) Trading Information

The blockchain browser also allows you to check the storage information of the file "test.jpg", TX HASH " 0x02e3216cc79528cd66286805b1ee5701e90ba4c6c4a6379ced8fe9ec64b2fe21", and the digital fingerprint of the stored file can be seen in TX DATA " 6674fcb2f5f3828edd19b3b077947592cc6be89f19cdc28d55e6e4f0634d484a" .As shown in Fig.11.

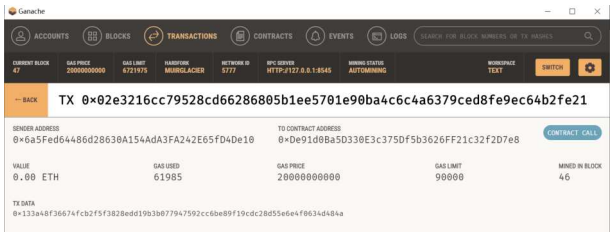


Fig. 11. Transaction information on the blockchain.

6) Analysis of results

Experimental test data were selected from 1M, 4M, 10M, and 50M files to test the performance of the execution of the deposit contract, by calling the contract to the platform and returning credentials to the user. Then through the credential information, forensic data query. The types of files tested include document files (doc, txt, pdf, ppt, xls), pictures (bmp,

jpg, png), audio (wav, mp3), video (mp4), etc., in which some of the types of files, file sizes, time of depositing and time of forensics are shown in Table II.

TABLE II. TEST DATA & RESULTS

File	*type*	*Size*	*store evidence*	*forensic*
test1	.docx	1288KB	00:01.28s	00:01.12s
test2	.jpg	1748KB	00:01.47s	00:01.40s
test3	.docx	4252KB	00:01.95s	00:01.76s
test3_2	.mp4	4285KB	00:02.03s	00:01.81s
test4	.mp4	10481KB	00:03.81s	00:03.51s
test5	.mp4	49728KB	00:13.79s	00:13.35s

After testing and comparison, it is found that the execution upload time of the contract, and the data query and forensic time both grow with the increase of the uploaded file, between the same type of file and different types of files. The file type does not have a significant effect on the time of depositing and forensics.

The SHA-256 algorithm chosen for the experiment has stable security and is also the digital fingerprint algorithm commonly used in electronic data authentication at present. The results of the deposit of all documents are verified by offline verification, and the results are consistent, which verifies the stability of the system.

This experimental environment is based on the official software provided by the Ethernet Foundation, or a general-purpose class library, which has good security and stability, and the development of the whole system is technically feasible.

V. SUMMARY

Blockchain electronic data deposition can standardize data deposition format, ensure data storage security, guarantee data flow traceability, effectively enhance the legal validity of evidence and court admissibility, and provide a more convenient service for users to obtain evidence and defend their rights. In this paper, we propose a model of an access evidence system and experimentally verify the technical feasibility and security of the deposited evidence forensic scheme, but many aspects need to be improved. First, the storage method of the original electronic evidence in the system can be considered in future work to adopt a decentralized storage architecture like IPFS to achieve. Second, the judicial process of electronic access to evidence can be further explored. From the extraction of evidence, the uploading of evidence, supervision, storage, demonstration, and cross-examination, the process can be further optimized.

ACKNOWLEDGMENT

This research was financially supported by the Scientific Research Project of Shan Dong University of Political Science and Law(2019Z01B), the Teaching Reform Project of Shandong University of Political Science and Law(2018JGA002), and Shandong Provincial Natural Science Foundation Major Basic Research (ZR2019ZD10).

REFERENCES

[1] GUO Shang-tong. WANG Rui-jin. ZHANG Feng-li. Summary of Principle and Application of Blockchain[J]. Computer Science, Vol.48,No.2, 2021(2):271-281. DOI:10.11896/jsjkx.200800021

[2] WANG Jian, ZHANG Yunjia, LIU Jiqiang, CHEN Zhihao. Blockchain-based Mechanism for Judicial Data Management and Electronic Evidence Storage[J]. Netinfo Security, 2022, 22(2): 21-31. DOI : 10.3969/j.issn.1671-1122.2022.02.003

[3] Jiongshu Chen. Research and Development of Digital Forensics System based on Distributed Ledger[D]. Chengdu, China: University of Electronic Science and Technology of China,2020

[4] Yang KunQiao. Research on key technologies of electronic storage and forensics based on blockchain [D].Chengdu, China: Sichuan University,2021

[5] Yang KunQiao. Research on electronic data deposit certificate platform based on blockchain [J]. Modern Computer, 2021(3): 36-40. DOI : 10.3969/j.issn.1007-1423.2021.09.007

[6] Lamprini Zarpala. Fran Casino. A Blockchain-based Forensic Model for Financial Crime Investigation: The Embezzlement Scenario[J].Digital Finance. Volume 3, Issue 3-4. 2021. PP 1-32. DOI : 10.1007/S42521-021-00035-5

[7] Xu Ronghong. Research on Electronic Evidence Storage Technology Based on Blockchain[D]. Beijing, China: North China University of Technology,2022

[8] Didi Cao Research and Implementation of Trusted Storage System Based on Blockchain and Smart Contract[D]. Nanjing, China: Nanjing University of Posts and Telecommunications,2019

[9] Lei Shang. A Review of Electronic Data Forensics Research Based on Blockchain Technology[J]. Office Informatization.2019,24(10):33-34

[10] Lu Yiming. Ding Shumeng. Jiang Shuhan. Fan Pengyi. Zhang Yunfan. Chen Yidan. Wang Qun. Research on electronic data deposition technology based on chain blockchain[J]. Network Security Technology & Application.2022(11): 118-121

2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA) | 978-1-6654-2490-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CAIBDA53561.2021.00035

Key Research on Recommendation Algorithms Based on Spatio-temporal Relationships in Location Social Networks

Qingbo Sun
Shandong University of Political Science and Law 250014
7466582536@dlvtc.edu.cn

***Abstract*—This article analyzes the location-based social network application model. The research content of this paper includes probability matrix model and convolution matrix model. The author studies the specific content of recommendation algorithms based on spatial relationships, recommendation algorithms temporal and spatial relationships. The author's purpose is to optimize the location social network environment and provide users with better service content.**

Keywords-location social network; probability matrix model; convolution matrix model

I. INTRODUCTION

In the context of the era of big data, people's social methods have also undergone major changes. Especially as the GPS technology system continues to mature, the popularity of location social networks is also increasing. At the same time, as the number of users and user needs continue to increase, location-based social networks are also facing problems such as excessive information and increased difficulty in screening. Using reliable algorithms to improve location social networks based on temporal and spatial relationships can not only increase the speed of information filtering, but also provide users with better information services, thereby satisfying users' social needs.

II. LOCATION SOCIAL NETWORK APPLICATION MODEL ANALYSIS

A. Probability Matrix Model

In the recommendation system, the traditional matrix factorization model MF mainly decomposes a high-dimensional sparse user item rating matrix into user potential feature vectors and item potential feature vectors in the shared potential feature space. Among them, the missing value in the matrix is calculated by the inner product of the potential feature vector of the user and the potential feature vector of the item. Assuming there are N users and M items, the observed scoring matrix is represented by R=(rij)M×N. Let U∈Rl×N and V∈Rl×M be denoted as the user potential feature matrix and the item potential feature matrix. Among them, the column vectors Ui and Vj respectively represent the potential feature vector of a specific user and the potential feature vector of a specific item, and the rating R of the item j by a specific user i is represented by the inner product of the corresponding potential vectors of the user i and the item j. The method of training the model is generally to minimize the loss function L (the loss function L refers to the sum of squares between the real score and the predicted score), and finally combine the L2 regularization term to deal with the overfitting problem.

B. Convolution Matrix Model

This model integrates CNN into PMF. Nowadays, the number and scale of users and projects in various e-commerce websites are huge and increasing, which leads to the problem of data sparseness in the scoring data of users on projects. This seriously affects the accuracy of the score estimation of the traditional collaborative filtering technology. In order to obtain the deep-level features of the text content, CNN technology is used, which has achieved great success in computer vision and natural language processing. It can effectively obtain the local features of pictures and text through modeling components. These model components include shared weights and down-sampling. It can enable a deeper understanding of the document. In addition, CNN can also generate in-depth understanding of review text through pre-trained word embedding models (such as Glove).

III. ANALYSIS OF RECOMMENDATION ALGORITHM BASED ON SPATIAL RELATIONSHIP

A. Basic Definition Analysis

Based on past experience, it can be understood that the index content involved in the analysis of the spatial relationship of the location social network includes user location indicators, scoring matrix relationships, user location relationships, location comment texts, etc. Suppose a set A, A={a1,a2,a3,...ai}, represents a set of i users, and a set B, B={b1,b2,b3,...bj}, represents a set of j locations. Based on this, a rating matrix is established, that is, R=(rmn)i×j, where rmn represents the rating of the mth user at the nth location. Q=(qmn)i×i represents the mutual relationship in the set of i×i locations. Combining the two types of relationships can analyze the comment text of the exploratory location association matrix. Researchers can obtain accurate analysis results to meet relevant application requirements.

B. Applied Algorithm Analysis

1) Conv MF Analysis

978-1-6654-2490-5/21/$31.00 ©2021 IEEE
DOI 10.1109/CAIBDA53561.2021.00035

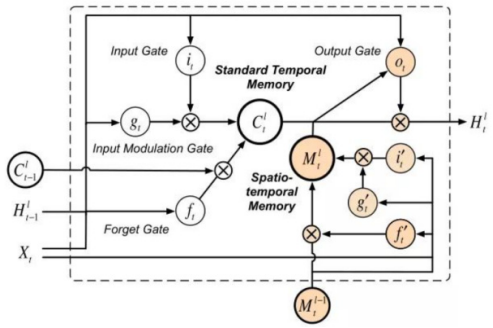


Figure 1. Conv MF Model Diagram

In the process of algorithm application, Conv MF model will be used as the basic framework in the analysis process. Analyzing from a probabilistic point of view requires inputting the obtained CNN data into the model for further sorting, and obtaining the required indicator information. From the perspective of actual application, the established Conv MF model will also use CNN information to complete document information collation, which improves the evaluation accuracy by 30%-50%. For example, in the process of project evaluation, implicit information (including quality information, type information, and quantity information) will also be sorted out. The system will match the details of the Conv MF model (as shown in Figure 1) to complete the data sorting. In this way, 20%-50% of the value data can be screened out for processing, and the reliability of the data analysis results can be improved.

2) CMFSR Model Analysis

In the process of parameter analysis, the CMFSR model is also used to complete data content sorting. In specific applications, assuming that the value of the geographic distance between the two sets of locations is less than the predetermined threshold, then it can be determined that there is a strong correlation between such related points. It can be understood from past model application experience that location-related content will be optimized in the model. Moreover, in the analysis process, its location-related network will be directly integrated into the established Conv MF model for further processing in use, and 30%-50% of value data can be extracted. In the analysis process of the incidence matrix, the system will also analyze its potential features, combined with the joint probability content. This can increase the accuracy of analysis by 40%-60% and improve the rationality of establishing network content.

C. Simulation Experiment Analysis

1) Data Set Analysis

In the established simulation experiments, the yelp website will be used as an important reference for the data set, so that it can provide the required information resources. Relying on the yelp website can not only collect basic user information, but also filter out 20%-50% of the effective data from user comments. The collected data also includes the real geographic location to meet the corresponding management needs. In order to improve the speed of data processing, the staff also need to simplify the processing of the original data. For example, extract and sort out high-quality customer reviews, and remove more than 30% of the original data. This can also improve the reliability of the data analysis results.

2) Experimental Process Analysis

From the specific application situation, the system needs to sort out the latent feature vector parameters (such as vector parameters), and design the dimensions of the parameters involved at 50. In the QC data set, at this time the normalization parameters a1=1, a2=100, a3=1, a0=50, and the ON data set can be divided into the following application parameters: a1=10, a2=20, a3 =5,a0=50. Moreover, the corresponding pre-processing work must be done in the production process of the comment text. Its content includes the following points. Firstly, clear the stop words and calculate the TF-IDF value of other information to obtain the parameter information for calculation. Secondly, according to the calculated results, the data whose frequency of occurrence exceeds 0.5 is cleaned up and the vocabulary content is further reduced on the original basis. At the same time, the system needs to select 5000-8000 groups of data with high frequency, and clean up the unconventional vocabulary in the past to obtain parameter information with application value.

D. Analysis of Results

1) Sorting out Experimental Results

The computer system will screen the relevant content based on the RMSE model based on the existing parameter content as an evaluation index. Moreover, the following collation results can be obtained based on the simulation experiment data. Firstly, the CMFSR model can be used to refine the results of data collation from the established data set. Meanwhile, various relationships will be sorted out in the framework system to improve the accuracy and refinement of the analysis results by 30%-50%. Secondly, on the basis of deep learning, the data content that has been collected is simulated. The computer system will be matched with CTR, CDL, and CMFSR models to obtain more reliable application features. This facilitates the smooth progress of the spatial relationship analysis work.

2) Spatial Relationship Analysis

Combined with the CMFSR model that has been built, it can be combined with the location relationship analysis situation to establish a spatial relationship matrix that meets the analysis. In the meantime, in the analysis process, researchers can also conduct a comprehensive analysis of the recommended results in the model. For example, in the CMFSR model, the scoring matrix calculation result G0=0, then the review file can be recommended at this time. If G0=∞, then it is necessary to analyze the potential characteristics of the location relationship of the review file at this time, so as to understand the rationality of the spatial relationship between each other. Simultaneously, in the CMFSR model analysis, researchers will also discuss the optimal results. Researchers need to combine the analysis results of the QC data collection and the ON data collection to obtain the best analysis results.

IV. Analysis of Recommendation Algorithm Based on Time-space Relationship

A. Basic Definition Analysis

Based on past experience, it can be understood that the index content involved in the analysis of the spatial relationship of the location social network includes user project ratings, social network information, project-related information, and comment text information. Assume that set A, A={a1, a2, a3,...ai}, represents a set of i users, and set B, B={b1, b2, b3,...bj}, represents a set of j items. Based on this, the item rating matrix is established, namely R=(rmn)i×j, where rmn represents the rating of the mth user on the nth item. Q=(qmn)i×i represents the mutual relationship within the set of i×i items, and accurate analysis results can be obtained by combining the two types of relationships [1].

B. Applied Algorithm Analysis

1) Conv MF Analysis

The Conv MF model is also used in the algorithm application process as the basic framework for analyzing the space-time relationship. From a probabilistic perspective, the computer system needs to input the obtained scoring matrix data into the model for further sorting, so as to obtain the feature vector parameters and improve the accuracy of the evaluation results by 30%-50%. For example, in the process of project evaluation, implicit vector information (including user feature vector information, item feature vector information, etc.) will also be sorted, and the data will be sorted with the details of the Conv MF model. The computer system needs to filter out 20%-50% of the value data from the established model for subsequent processing, so as to improve the scientificity and usability of the data analysis results [2].

2) JCMF Model Analysis

The JCMF model is also used in the process of spatio-temporal parameter analysis to complete data content sorting. In specific applications, assuming that the value of the relationship between the two sets of items is less than the established threshold, then the model can be used as the basic content to carry out the correlation analysis. It can be understood from past model application experience that location-related content will be optimized in the model. In the analysis process, its location association network will be directly integrated into the established JCMF model for further processing in use, and 20%-50% of the value data will be extracted. Subsequently, the computer system completes the data sorting with the help of the JCMF model. The use of vector feature analysis results can improve the analysis accuracy of 30%-45% based on the joint probability content, thereby improving the rationality of the established network content [3].

C. Simulation Experiment Analysis

1) Data Set Analysis

The computer system will also use the yelp website to sort out the data that needs to be analyzed in the established simulation experiment, and then obtain the required data set. Moreover, the yelp website can not only be used to collect basic user information, but also 20%-50% of reliable data can be filtered from the classified items. Besides, the actual geographic location is also included in the collected data to meet the corresponding analysis and processing needs. In order to increase the speed of data sorting, the computer system also needs to simplify the processing of the original data, and optimize and sort the interfering data information. In this way, the reduction rate can be controlled at 30%-50%, which also improves the application value of data information [4].

2) Experimental Process Analysis

From the specific application situation, the computer system needs to sort out the latent eigenvector parameters, and design the dimensions of the parameters involved at 50. In the NC data set, the regularization parameters a1=15, a2=10, a3=0.1, a4=70, a0=150 at this time. The WI data set can be divided into the following application parameters: a1=10, a2=7, a3=0.001, a4=70, a0=80. In the OH data set, it can be divided into the following application parameters: a1=1, a2=100, a3=0.0001, a4=70, a0=70. Researchers also need to do the corresponding pre-processing work in the process of making comment texts. The specific content can be referred to the chapter above to obtain the parameter information with application value [5].

D. Analysis of Results

1) Sorting out Experimental Results

The computer system will screen the value data in the NC, WI, and OH data sets based on the RMSE model based on the existing parameter content and sort them out. The summary results obtained are as follows. Firstly, the computer system can use the JCMF model to refine the results of data sorting from the established data collection. Simultaneously, the project relationship and user management will be sorted out in the established structure system to improve the accuracy of the analysis results by 30%-50%. Secondly, the computer system can simulate the collected data content based on the neural network recommendation method. The computer system is matched with the MF recommendation algorithm to obtain more effective application data, so as to facilitate the smooth progress of the latent feature analysis work [6].

2) Project Association Analysis

Combining the JCMF model that has been built, and the project relationship analysis situation can build a project relationship matrix that satisfies the analysis. At the same time, in the analysis process, the computer system will also conduct a comprehensive analysis of the recommended results in the model. For example, in the JCMF model, the scoring matrix calculation result Gc=0, then the item relationship can be recommended at this time. If Gc=∞, then it is necessary to analyze the potential characteristics of the relationship between the items in the review content, so as to understand the rationality of the spatial relationship between the items. In addition, in the analysis of the JCMF model, the relevant content of the recommended results of the JCMF model will be sorted out. Moreover, the computer system will also analyze the significance of the project relationship. Based on the obtained parameter information, the calculated value $p<0.01$, so that better analysis results can be obtained to meet the analysis needs [7].

3) Convergence Analysis

The computer system also needs to complete the convergence analysis based on the JCMF model to understand the reliability of the model's convergence performance. For example, the number of iterations set in the JCMF model parameter experiment process is 300, and the error content is controlled within a small range. In this way, the early stopping method can be used more for processing to obtain reliable data analysis results. In the process of model analysis, user relationships and project relationships will also be fully integrated to obtain reliable convergence analysis results. According to the data information display results, it can be shown that relying on the convergence content of the JCMF model, its work efficiency can be increased by 30%-50%, meeting application requirements [8].

V. Conclusion

In summary, a large amount of application data needs to be collected when performing application analysis based on location social networks. These data include user comment information, social comment information, and geographic parameter information. Researchers need to match appropriate application models such as JCMF model, Conv MF model, CMFSR model to assist the analysis work when conducting comprehensive analysis, and to summarize and store the analysis data. This makes it easy to obtain reliable analysis results and meet the needs of model applications.

References

[1] Ding Jinjie, Xie Weixin, Li Yongfeng, Huang Zitong. Correlation filtering and tracking algorithm focusing on learning spatio-temporal relationships [J/OL]. Signal processing: 1-13 [2021-05-01].

[2] Fan Yuqi, Liu Yulan, Xu Xiong, Guo Dan, Wen Pengfei. Radar target track recognition based on point-track time-space relationship[J]. Journal of Electronic Measurement and Instrument, 2020, 34(09): 108-116.

[3] Zhu Qing, Feng Bin, Li Maosu, Chen Meite, Xu Zhaowen, Xie Xiao, Zhang Yeting, Liu Mingwei, Huang Zhiqin, Feng Yicong. Efficient sparse graph indexing method for dynamically associated data[J]. Journal of Surveying and Mapping, 2020, 49 (06):681-691.

[4] Liu Yun, Wang Ziyu. Research on optimization of spatio-temporal causality with non-parametric trigger algorithm[J]. Journal of Yunnan University (Natural Science Edition), 2019, 41(06): 1130-1136.

[5] Wang Yuankui, Qin Bo, Li Wei. Real-time detection of urban road vehicles based on time-space relationship model[J]. Computer System Applications, 2017, 26(10): 207-212.

[6] Li Min, Chen Yanping, Song Ricong. The description method of the time-space relationship of the moving human body based on the 13-tuple model[J]. Journal of Mianyang Normal University, 2017, 36(05): 73-79.

[7] Jiang Zhihao, Li Minxian, Zhao Chunxia, Shao Qingwei. Pedestrian activity prediction algorithm based on spatio-temporal model video surveillance [J]. Computer Applications and Software, 2017, 34(01): 149-153.

[8] Zhang Xueping, Li Weicheng, Zhu Yuhua. Research on spatiotemporal association rule mining algorithm based on FP-tree [J]. Microelectronics and Computer, 2016, 33(08): 130-133+138.