Authorized Protection Mechanism of Software Cloud Based on Time-Specific Encryption

Ke Yuan, Junyi Wu, Baolei Zhang, Bozhen Wang, Yuye Wang*, Member, IAENG, and Chunfu Jia

***Abstract*—In order to improve the security, ease of use and flexibility of software products, this paper proposes a software cloud authorized protection mechanism based on specific time encryption. This mechanism sends the encrypted and packed executable file to the cloud server, so that only legitimate users can unpack, decrypt and run the software in the cloud within a specific time interval. On this basis, this article gives a formal definition of the solution model and constructs a specific implementation solution based on the random oracle model. Security analysis shows that this solution can not only prevent attackers from using cloud servers and time servers to illegally obtain software usage rights, but also reduce time overhead and costs, thereby effectively protecting the legitimate rights and interests of developers and users.**

***Index Terms*—time-specific encryption; authorized protection; software packing; time trapdoor; time server; cloud server.**

I. Summary

As enterprise and individual users have increasingly diversified software needs, more and more software adopts the model of purchasing usage rights on demand, such as based on usage duration, number of times, and frequency. This model not only effectively lowers the price threshold of software, but also significantly improves user acceptance and satisfaction. Professional big data analysis software, including IBM SPSS and SAS, has gradually shifted to a pay-as-you-go model, in line with market development trends.

In recent years, there has been a growing emphasis on individual health, resulting in the rapid advancement of smart wearable devices. As a result, the smart wearable companies constitute a segment of the consumer base for big data analysis software.

Manuscript received November 14, 2023; revised July 31, 2024. This work was supported by the National Natural Science Foundation of China under Grant 61972215, 61972073 and 62172238; the Natural Science Foundation of Tianjin under Grant 20JCZDJC00640; the Key Research and Promotion Projects of Henan Province under Grant 222102210062 and 232102210071; the Basic Research Plan of Key Scientific Research Projects in Colleges and Universities of Henan Province under Grant 22A413004; the Innovation Training Program for College Students of Henan province under Grant 202310475143.

Ke Yuan is an Associate Professor of School of Computer and Information Engineering, Henan University, Kaifeng, 475004, China; Henan Province Engineering Research Center of Spatial Information Processing, Henan University, Kaifeng, 475004, China (e-mail: yuanke@henu.edu.cn).

Junyi Wu is a postgraduate student of School of Computer and Information Engineering, Henan University, Kaifeng, 475004, China (e-mail: WuJunyi@henu.edu.cn).

Baolei Zhang is a postgraduate student of College of Cybersecurity, Nankai University, Tianjin, 300350, China (e-mail: zhangbaolei@mail.nankai.edu.cn).

Bozhen Wang is an undergraduate student of International Business School, Henan University, Zhengzhou, 450046, China (e-mail: WangBozhen@henu.edu.cn).

Yuye Wang is an Associate Professor of School of Computer and Information Engineering, Henan University, Kaifeng, 475004, China (corresponding author to provide e-mail: wangyuye@henu.edu.cn).

Chunfu Jia is a Professor of College of Cybersecurity, Nankai University, Tianjin, 300350, China (e-mail: cfjia@nankai.edu.cn).

As a novel form of human-computer interaction, smart wearable devices provide consumers with exclusive and personalized services and collect a variety of human data by wearing smart devices on the human body. Currently, wearable devices like smartwatches have demonstrated their ability to detect and monitor the progression and treatment of various diseases in their early stages through biophysical signals. Enterprises utilize smart wearable devices to collect user health data, monitor treatment processes, and offer real-time feedback in order to deliver optimized treatment plans and enhance treatment effectiveness[1, 2].

To achieve the aforementioned functions, enterprises need to analyze and process the data collected by smart wearable devices. While large companies like Fitbit, Garmin, and Apple may have their own big data analytsis platforms for processing and analyzing health data from their smart wearable devices, small and medium-sized smart wearable device enterprises may opt to use a third-party big data analysis platform due to technical and resource limitations. This approach allows them to reduce operating costs by paying a fee for the usage time of the software, rather than building and maintaining a complete big data analysis platform themselves.

Nevertheless, the method of payment for accessing software usage duration is susceptible to unauthorized access. Hence, it is imperative for software developers to recognize the significance of security concerns and implement protective measures to safeguard the integrity of their software.

Currently, software protection is categorized into hardware-based and software-based protection. Hardware-based protection commonly relies on dongles, encryption locks, and their drivers, but may encounter usability issues. Software-based protection involves using registration codes and license files to compare and verify original data by accessing a dedicated server database in a networked environment[3]. However, registration codes[4] and license files only offer basic security guarantees and are susceptible to reverse engineering attacks. Moreover, dedicated servers escalate hardware, manpower, and operational and maintenance management costs for software developers, with expenses growing exponentially as user groups expand. Therefore, conventional software protection fails to meet the requirements of secure online distribution and on-demand sales of software.

Cloud servers (CSs) have the characteristics of strong computing power, high reliability, flexibility, and ease of use[5]. With the rise of the Software as a Service (SaaS) model, more and more software has begun to be provided to users in the form of online services, which greatly facilitates user use and management. In the SaaS model, software applications are hosted on cloud servers and provide services to users through the Internet. Users do not need to install or

maintain software, and only need a computer loaded with the Internet to obtain related services. Software developers can easily share software resources, and at the same time, the investment in hardware facilities, human resources and operation and maintenance costs is greatly reduced.

Building upon the aforementioned, the research objective of this paper is to propose a Time-Specific Encryption (TSE)-based software authorized protection in the cloud (TSE-SAPC). This approach leverages the comprehensive application of cloud computing, public key, and block cipher encryption technology. It enables users to be authorized to use the software within specific time intervals tailored to their actual needs, while restricting access during other periods. The scheme exhibits robust security, effectively thwarting attackers from exploiting CS and time server (TS) to gain unauthorized software usage rights. Additionally, it boasts the advantages of cost-effectiveness, convenient operation and maintenance, ease of use and management, all while ensuring the legitimate rights and interests of both developers and users are considered.

II. Basic knowledge

A. The concept of BDH

At present, public key cryptographic algorithms are mainly based on the large integer factorization problem (IFP)[6, 7], the discrete logarithm problem (DLP)[8] of multiplication group over finite fields, and the elliptic curve discrete logarithm problem (ECDLP)[9] over finite fields. By using bilinear pairings, ECDLP over finite fields can be reduced to the DLP of the multiplication group over a finite field. The specific scheme proposed below is based on bilinear pairwise construction, and its security is grounded in the bilinear Diffie-Hellman (BDH)[10–12] problem. The related knowledge is outlined in the sequel.

Definition 1 Bilinear mapping. Suppose G_1 is an elliptic curve discrete logarithm addition group over a finite field, G_2 is discrete logarithm multiplication group over a finite field, and the orders of G_1 and G_2 are prime numbers denoted by Q. The mapping $e : G_1 \times G_1 \rightarrow G_2$ is termed bilinear mapping and is required to satisfy the following attributes:

a) Bilinear property: For any $P, Q, R \in G_1$, there are $e(P + Q, R) = e(P, R)e(Q, R)$ and $e(P, Q + R) = e(P, Q)e(P, R)$.

b) Non- degeneracy: If P is the generator of G_1, $e(P, P)$ is the generator of G_2.

c) Calculability: For any $P, Q \in G_1$, there is an algorithm to calculate $e(P, Q)$.

Definition 2 BDH. Let P be the generator of G_1. Given a tuple (P, aP, bP, cP), it is difficult to calculate $e(P, P)^{abc}$ for any $a, b, c \in \mathbb{Z}_q^*$.

The security proof in this paper relies on the assumption of the computational difficulty of solving the BDH problem.

B. TRE and TSE concepts

The cryptography primitive Timed-Release Encryption (TRE)[13], involves the sender transmitting an encrypted message containing information about the "future" to the receiver. The receiver can only decrypt the message when a specific time is reached. TRE was initially proposed by May[14] in 1993 and further explored by Rivest et al.[15] in 1996. Subsequently, its theory and applications have significantly evolved[16–19]. In the existing TRE implementation methods[20–25], the time trapdoor is established through the periodic broadcast of the TS. If the user fails to receive or loses the time trapdoor, the message cannot be decrypted. Recognizing this limitation, in 2010, Paterson et al.[15] introduced the concept of Time-specific Encryption (TSE) to decrypt messages within a specific time interval. TSE enables the receiver to decrypt messages by using any time trapdoor broadcast by the TS in that interval[26]. Paterson et al. formulated the standard TSE scheme by adopting the time binary tree and identity-based encryption technology [27, 28]. They extended this to create the Public Key TSE (PK-TSE) and Identity-Based TSE (ID-TSE) schemes. In 2012, Kasamatsu et al. proposed a TSE scheme based on forward secure encryption [29–31]. Building on the TSE design concept, subsequent researchers have introduced application schemes for specific time encryption in various scenarios, including pay TV services[32, 33], online competitions[34], time locks[35, 36], and certificateless encryption[37].

C. Software packing concept

A shell is a piece of code designed to protect software from illegal modification or decompilation[38]. Once the software is encapsulated within a shell, attackers must first break through the shell to obtain the "pure" version of the executable file. Therefore, the shell plays a crucial rule in safeguarding the software. Software shells are generally categorized into encryption and compression shells. The encryption shell encrypts and decrypts the PE format of the executable file to achieve anti-debugging, anti-tracking and other protection functions. The compression shell mainly plays the role in compressing the size of the program and reducing the space occupation. This paper adopts the encryption shell mode.

In this paper, the encryption shell approach is used. Specifically, when the encrypted executable file is loaded into memory, the shell program code first obtains program control, executes decryption to restore the executable file source code, and finally continues executing the original code.

III. TSE-SAPC model

Assuming that the code segment of the original executable (OEF) is SM4-encrypted with the SM4[39] encryption key MK on the software developer terminal (Te), the resulting executable file (PEF) with the encryption shell is generated and transmitted to the CS. The system's design objective is to enable user (R) to securely and efficiently utilize the software within a time chain consisting of multiple specific intervals. In this paper, this model is referred to as the authorized protection model of software cloud based on Time-Specific Encryption, as illustrated in Figure 1, the model works as follows:

1) The software developer encrypts the code segment of the original executable file OEF to obtain the PEF, and deploys the PEF in the cloud server.

2) The software developer generates its public and private key pair (pk, sk) for the user.

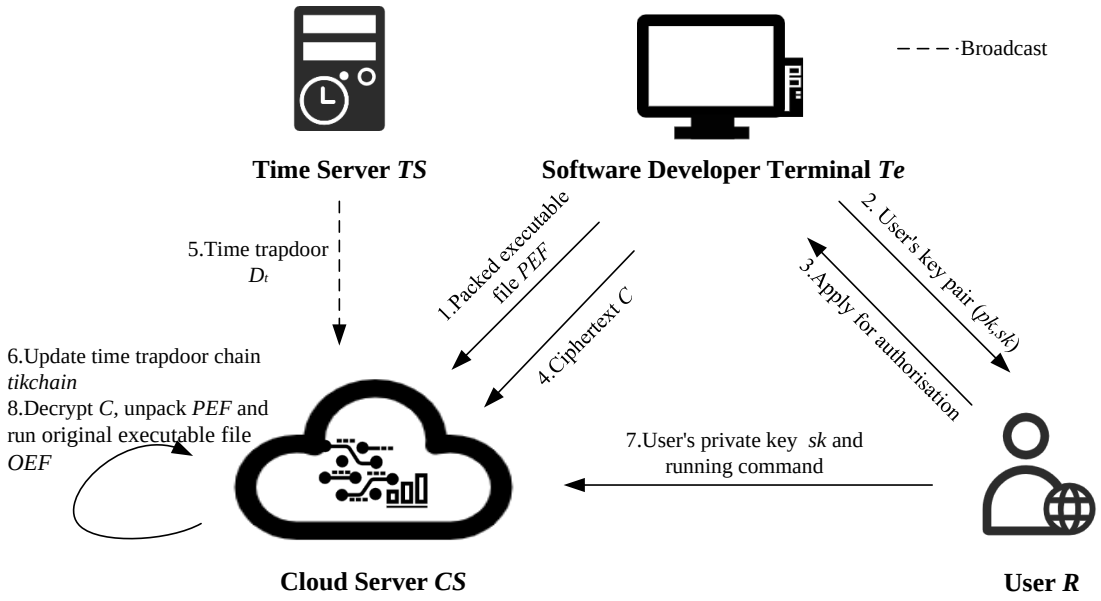

Fig. 1: Authorized protection model of software cloud based on time-specific encryption

3) The user pays a certain fee to the software developer to apply for software usage rights according to their needs.

4) The software developer generates ciphertext C that can be decrypted only during the authorized period for authorized user, and uploads it to the cloud server.

5) The time server broadcasts the time trapdoor D_t upon reaching time t.

6) The cloud server receives the time trapdoor and updates the time trapdoor chain $tikChain$.

7) The user establishes a secure connection with the cloud server within the authorized time period and sends the private key sk and running command to the cloud server.

8) The cloud server runs OEF after successfully unpack PEF through $sk, C, tikChain$.

The formal definition is outlined next.

Definition 3 TSE-SAPC model. It encompasses four entities: TS, CS, Te and R, and algorithm 8-tuples $\xi_{\text{TSETIK-STUC}} = (MKTIK\text{-}Setup, SM4.Enc, PK.Setup, PK.KeyGen, PK.Enc, PK.TIK\text{-}Ext, TIK\text{-}Chain, SM4TIK\text{-}PK.Dec)$. The algorithm 8-tuples $\xi_{\text{TSETIK-STUC}}$ are as follows:

MK-Setup. Executed by software developers. An encryption key $MK = (MK_0, MK_1, MK_2, MK_3)$ is generated on Te, where $MK_i\,(i = 0, 1, 2, 3)$ is a 32-bit value.

SM4.Enc. Executed by software developers. MK is used to encrypt the code segment of the OEF on Te with SM4. The executable file $PEF = SM4.Enc(MK, OEF)$ is generated with the encryption shell and transmitted to the CS.

PK.Setup. Executed by the time server. The security parameter k and the total time length L are input to generate the public and private key pair $(TS\text{-}MPK, TP\text{-}MSK)$ of the TS.

PK.KeyGen. Executed by software developers. The security parameter k is input on Te to generate the public and private key pair (pk, sk) for the software user.

PK.Enc. Executed by software developers. MK is encrypted by SM4 with parameters such as $TS\text{-}MPK, [t_0, t_1], pk$, and the ciphertext $C = PK.Enc(TS\text{-}MPK, [t_0, t_1], pk, MK)$ can only be decrypted within the time interval $[t_0, t_1]$.

PK.TIK-Ext. Executed by the time server. Represent time t with the string $T \in \{0,1\}^*$. Parameters such as $t, TS\text{-}MPK, TS\text{-}MSK$ are used to generate the corresponding time trapdoor D_t through an independent signature scheme, and D_t is broadcasted.

TIK-Chain. Executed by the cloud server. The time trapdoor *Dt* is connected into the time trapdoor chain $tikChain$.

SM4-PK.Dec. If user R uses the software within the time interval $[t_0, t_1]$, they need to send the running PEF command and their own private key sk to cloud server CS; The CS decrypts the PEF with C, sk, and $tikChain$, unpacks the code segment of the original executable file OEF, namely $SM4\text{-}PK.Dec(PEF, C, TS\text{-}MPK, tikChain, sk)$, and then runs the OEF.

IV. TSE-SAPC Structural scheme

This section presents a concrete TSE-SAPC construction scheme and provides a security analysis.

A. Scheme expression

Based on the previously defined bilinear pairings and BDH problem, a specific TSE-SAPC solution is formulated. The work process involves the following 8 stages:

MK-Setup. Initialize the SM4 encryption key MK: randomly select a number and assign it to $MK_{[i]} \in \mathbb{Z}_2^{32}\,(i = 0, 1, 2, 3)$ to generate the SM4 encryption key MK.

SM4.Enc. Software developers perform the following operations:

a) Use MK to encrypt the code segment of the original executable file OEF with SM4;

b) Execute the packing operation, change the entry address of the OEF to the entry address of the shell code.

PK.Setup. Given a security parameter k and the total time length L, output the system parameters $params = \{k, q, G_1, G_2, e, P, H_1, H_2, n\}$, the public-private key pair $(TS\text{-}MPK, TS\text{-}MSK)$ of the TS, and the complete binary tree with depth $d\left(\mathrm{L} = 2^d\right)$, as shown in Figure 2. G_1 is an additive group , G_2 is a multiplicative group, both with prime orders q, and the mapping $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear mapping satisfying Definition 2, and $H_1 : \{0,1\}^* \rightarrow G_1, H_2 : G_2 \rightarrow \{0,1\}^n$ are hash functions with n being the plaintext length. TS randomly selects a generator $G \in G_1^*$, and selects a random number $TS\text{-}MSK = s \in \mathbb{Z}_q^*$ as the private key, which corresponds to public key $TS\text{-}MPK = (G, sG)$. $params$ and $TS\text{-}MPK$ are public parameters.

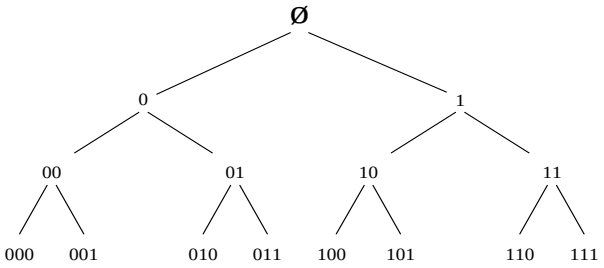

Fig. 2: Full binary tree with depth 3

PK.KeyGen. The public parameter is taken as input, a random number $sk = u \in \mathbb{Z}_q^*$ is selected as the user's private key, and the user public key $pk = (uG, usG)$ is computed.

PK.Enc. Given the SM4 key MK, the time server's public key $TS\text{-}MPK = (G, sG)$, the user's public key pk and the time interval [t_0,t_1] during which the user uses the software, the software developer performs verifies if $e(uG, sG) = e(G, usG)$; only when the equation holds, the following operations are performed:

a) Generate the minimum root set $S_{[t_0,t_1]}$ of subtrees covering all time nodes of $[t_0, t_1]$. Taking the tree shown in Figure 2 as an example, suppose that the software purchased by Bob is used for a period of $[0, 5]$, then $S_{[0,5]} = \{0, 10\}$. For each $y \in S_{[t_0,t_1]}$, select $r \in \mathbb{Z}_q^*$ randomly and calculate rG and $rusG$, then calculate

$$K = e\left(rusG, H_1\left(y\right)\right) = e\left(G, H_1\left(y\right)\right)^{rus}$$

b) Get the ciphertext set

$$CT_{[t_0,t_1]} = \{c_y = \langle U, V\rangle = \langle rG, MK \oplus H_2\left(K\right)\rangle : y \in S_{[t_0,t_1]}\}$$

and generate the ciphertext $C = \left(CT_{[t_0,t_1]}, [t_0, t_1]\right)$.

PK.TIK-Ext.At the arrival of time t, the time server generates the set ρ_t of nodes in the path from the root node to node $t$in the binary tree. Taking Figure 2 as an example, if $t = 3$ at this time, the corresponding path set $\rho_t = \{\varnothing, 0, 01, 011\}$. Subsequently, the time server publishes the time trapdoor $D_t = \{d_x = sH_1\left(x\right) : x \in \rho_t\}$. Every user can verify its authenticity, i.e.,

$$Sign\left(x\right) : d_x = sH_1\left(x\right)$$

$$Ver\left(d_x, x\right) : e\left(sG, H_1\left(x\right)\right) = e\left(G, sH_1\left(x\right)\right)$$

where $Sign\left(\right)$ is the signature function and $Ver\left(\right)$ is the verification function.

TIK-Chain. Taking the time trapdoor chain $tikChain$ and the time trapdoor D_t issued by the time server as input, the cloud server generates a linked list node and links it to the end of the $tikChain$ to generate a new $tikChain$. Figure 3 shows the tikChain generated by the cloud server at the current time $t = 3$.

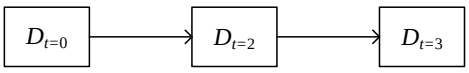

Fig. 3: Time trapdoor chain at current time $t = 3$

SM4-PK.Dec. Given the packed executable file PEF, user's private key sk, user's corresponding SM4 encryption key ciphertext $C = \left(CT_{[t_0,t_1]}, [t_0, t_1]\right)$, and time trapdoor chain *tikChain*, the CS performs the following operations:

a) Retrieve the data of the tail node of $tikChain$ to obtain the time trapdoor D_t corresponding to the current time t.

b) Generate the minimum root set $S_{[t_0,t_1]}$ of subtrees covering all time nodes of [t_0,t_1] and the set ρ_t of nodes in the path from the root to node t. Calculate the unique element z where ρ_t and $S_{[t_0,t_1]}$ intersect.

c) Calculate $K' = e\left(U, d_z\right)^u$.

d) Calculate $V \oplus H_2\left(K'\right)$, restore the SM4 encryption key MK. If C is the correct ciphertext, then $U = rG, V = MK \oplus H_2\left(K\right)$, in which $K = e\left(G, H_1\left(z\right)\right)^{rus}$, $z \in S_{[t_0,t_1]}$. The decryption correctness is verified as follows:

$$\begin{aligned} K' &= e(U, d_z)^u \\ &= e(rG, sH_1\left(z\right))^u \\ &= e(G, H_1\left(z\right))^{rus} \\ &= K \end{aligned}$$

$$\begin{aligned} V \oplus H_2\left(K'\right) &= MK \oplus H_2\left(K\right) \oplus H_2\left(K'\right) \\ &= MK \end{aligned}$$

e) The shell program of the file PEF reads the ciphertext of the code segment of the OEF and uses the MK decrypted by the above operation d) to decrypt the code segment cipher. The shell program of PEF returns the execution right to the OEF after successful decryption.

B. Security analysis

The proposed TSE-SAPC scheme sets the cloud server CS, the time server TS, and the legitimate user R to be honest but curious. Each entity faithfully provides services in accordance with the defined rules without engaging in collusion. However, all entities have the incentive to attempt illegal acquisition of the right to use the software based on the information they have gathered. In our scheme, the process involves decrypting the encrypted key through the cloud server and subsequently running the shell software.

To mitigate the risk of unauthorized use of the software and the leakage of the software encryption key by the cloud server (CS), we assume that the software developer has implemented a comprehensive audit mechanism that regularly audits and monitors software usage on the cloud server. We also assume that during the contract signing between the software developer and the cloud service provider, clear definition of software usage rights and restrictions are outlined. Consequences and responsibilities for unauthorized use of the software are also specified. Additionally, the scheme considers potential threats from malicious attackers who may attempt to reversely crack the code of the shelled executable file or launch violent attacks to crack the private key of the legitimate user, thus illegally obtaining the right to use the software. The ensuing security analysis addresses these threats and demonstrates that the TSE-SAPC scheme is secure against such potential risks.

Theorem 1. This scheme can prevent the time server from obtaining the right to use the software.

Proof. Time server TS acts as the producer of time trapdoors and and possesses all the time trapdoors generated and broadcast according to the rules. The SM4 key is encrypted by the TS public key $TS\text{-}MPK$, the user's public key pk, and the decryptable time interval $[t_0, t_1]$ before transmission.

In this scheme, the user's public key $pk = (uG, usG)$ is known. According to ECDLP, given G and sG, it is difficult to calculate u, which ensures the safety of the user's private key $sk = u \in \mathbb{Z}_q^*$.

As there is no access to the user's private key, the TS encounters difficulty in decrypting the ciphertext of the SM4 key to execute the OEF on the CS. This difficulty is tantamount to breaking a public key cryptography mechanism, making it evidently challenging. The theorem is thus proven.

Theorem 2. This scheme can prevent malicious attackers from gaining software usage rights.

Proof. Malicious attackers may interact with the cloud server CS. In this scheme, the SM4 key needs to be encrypted through the TS public key $TS\text{-}MPK$, the user public key pk, and decryptable time interval $[t_0, t_1]$ before transmission, and relying on the difficulty of DLP for security. As the malicious attacker lacks access to the legitimate user's private key, decrypting the ciphertext of SM4 key to execute the original executable file on the CS becomes arduous. This challenge equates to breaking a public key cryptography mechanism and is obviously very difficult.

Furthermore, if a malicious attacker intends to obtain the shelled executable file for reverse cracking, they must first attack the CS to illegally obtain the shelled executable file, and subsequently attack the SM4 encryption mechanism. The complexity of this process is notably high, substantiating the theorem.

Theorem 3. This scheme can prevent legitimate users from using the software before and after a specific time interval.

Proof. The legitimate user R possesses their private key and all time trapdoors broadcast by the TS. Before transmitting the SM4 key, it must be encrypted by the TS public key $TS\text{-}MPK$, the user public key pk and the decryptable time interval $[t_0,t_1]$. Since the time t corresponding to the time trapdoor of the end node of the time trapdoor chain on the current cloud server does not belong to the decryptable time interval $[t_0,t_1]$, therefore, when the user sends the running command to the CS in an unauthorized time interval, the CS cannot decrypt the $SM4$ key ciphertext to run the OEF. This challenge mirrors breaking a public key cryptography mechanism, confirming the validity of the theorem.

V. Conclusion and future work

After conducting an in-depth study of TSE and its extended schemes, this paper introduces a novel specific time encryption scheme, TSE-SAPC, focusing on software authorization protection. Leveraging the block cipher algorithm SM4 and BDH double encryption technology, TSE technology and software shell technology, the paper provides a formal definition of scheme model and outlines the scheme construction method based on the random Oracle model. A comprehensive security analysis is also presented. The key advantages include: ①Legal users can pay according to the usage time, allowing for periodic and aperiodic use, significantly reducing user costs; ②The inclusion of the TS and the introduction of the time trapdoor chain mitigate the risk of collusion between the CS and illegal users attempting to bypass time restrictions; ③ On the basis of the security provided by the CS, packing operation further improves the security of the software, and the executable files are started on the CS after packing, which reduces the possibility of illegal users to reverse analyze the software; ④The integration of symmetric and asymmetric cryptography in a dual encryption scheme further strengthens software security.

With the rapid development of cloud computing, the cloud operation mode of software has become more and more popular. Based on the cloud operation mode of software, this scheme provides developers with a new way to sell software securely and provides users with a new way to use software periodically. However, the security of the scheme proposed in this paper depends on the security of the cloud server. In order to prevent cloud servers from illegally using software or leaking software encryption keys, software developers need to establish a complete audit mechanism or choose a secure and reliable cloud service provider. In future work, we will explore how to improve the accuracy of software usage time limits by increasing the frequency of time server broadcast time traps, and design a scheme to achieve software cloud security protection in a completely untrusted environment.

References

[1] M. A. Jan, W. Zhang, F. Khan, S. Abbas, and R. Khan, "Lightweight and smart data fusion approaches for wearable devices of the internet of medical things," *Information Fusion*, vol. 103, p. 102076, 2024.

[2] D. Nahavandi, R. Alizadehsani, A. Khosravi, and U. R. Acharya, "Application of artificial intelligence in wearable devices: Opportunities and challenges," *Computer Methods and Programs in Biomedicine*, vol. 213, p. 106541, 2022.

[3] L. Keqiu, S. Xiaolong, and X. Junfeng, "An android software protection scheme based on anti-debugging and online encryption verification," *Transactions of Beijing Institute of Technology*, vol. 37, no. 12, pp. 1276–1281, 2017.

[4] C. Jinyu and H. Chuqi, "Research on software copyright protection mode based on machine feature registration code," *Journal of Chongqing University of Technology: Natural Science*, vol. 33, no. 1, pp. 125–129, 2019.

[5] F. Khoda Parast, C. Sindhav, S. Nikam, H. Izadi Yekta, K. B. Kent, and S. Hakak, "Cloud computing security: A survey of service-based models," *Computers & Security*, vol. 114, p. 102580, 2022.

[6] P. L. Montgomery, "A survey of modern integer factorization algorithms," *CWI quarterly*, vol. 7, no. 4, pp. 337–366, 1994.

[7] K. Djebaili and L. Melkemi, "A different encryption system based on the integer factorization problem," *Malaysian Journal of Computing and Applied Mathematics*, vol. 3, no. 1, pp. 37–41, 2020.

[8] K. S. McCurley, "The discrete logarithm problem," in *Proc. of Symp. in Applied Math*, vol. 42. USA, 1990, pp. 49–74.

[9] S. Ullah, J. Zheng, N. Din, M. T. Hussain, F. Ullah, and M. Yousaf, "Elliptic curve cryptography; applications, challenges, recent advances, and future trends: A comprehensive survey," *Computer Science Review*, vol. 47, p. 100530, 2023.

[10] Y. Yacobi, "A note on the bilinear diffie-hellman assumption." *IACR Cryptol. ePrint Arch.*, vol. 2002, p. 113, 2002.

[11] M. Ebina, J. Mita, J. Shikata, and Y. Watanabe, "Efficient threshold public key encryption from the computational bilinear diffie-hellman assumption," in *Proceedings of the 8th ACM on ASIA Public-Key Cryptography Workshop*, 2021, pp. 23–32.

[12] P. Wuttidittachotti and P. Natho, "Improved ciphertext-policy time using short elliptic curve diffie-hellman." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 13, no. 4, 2023.

[13] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," *Massachusetts Institute of Technology*, 2001.

[14] W. Mao, "Timed-release cryptography," in *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8*. Springer, 2001, pp. 342–357.

[15] K. G. Paterson and E. A. Quaglia, "Time-specific encryption," in *International Conference on Security and Cryptography for Networks*. Springer, 2010, pp. 1–16.

[16] Y. ke, L. Zheli, J. chunfu, M. Haoyu, and L. Shuwang, "Research on timed-release encryption," *Journal of Computer Research and Development*, vol. 51, no. 6, pp. 1206–1220, 2014.

[17] Y. Ke, L. Zheli, J. Chunfu, Y. Jun, and L. Shuwang, "Public key timed-release searchable encryption in one-to-many scenarios," *Acta Electronica Sinica*, vol. 43, no. 4, p. 760, 2015.

[18] Y. Ke, W. Zilin, D. Zhanfei, H. Xinzheng, J. Chunfu, and H. Yuan, "Anonymous query mechanism construction of timed-release encryption." *Advanced Engineering Science/Gongcheng Kexue Yu Jishu*, vol. 54, no. 3, 2022.

[19] K. Yuan, H. Cao, S. Zhang, C. Zhai, X. Du, and C. Jia, "A tamper-resistant timed secure data transmission protocol based on smart contract," *Scientific Reports*, vol. 13, no. 1, p. 11510, 2023.

[20] J. Cathalo, B. Libert, and J.-J. Quisquater, "Efficient and non-interactive timed-release encryption," in *Information and Communications Security: 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005. Proceedings 7*. Springer, 2005, pp. 291–303.

[21] A.-F. Chan and I. F. Blake, "Scalable, server-passive, user-anonymous timed release cryptography," in *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. IEEE, 2005, pp. 504–513.

[22] S. S. Chow, V. Roth, and E. G. Rieffel, "General certificateless encryption and timed-release encryption," in *Security and Cryptography for Networks: 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings 6*. Springer, 2008, pp. 126–143.

[23] G. Choi and S. Vaudenay, "Timed-release encryption with master time bound key," in *Information Security Applications: 20th International Conference, WISA 2019, Jeju Island, South Korea, August 21–24, 2019, Revised Selected Papers 20*. Springer, 2020, pp. 167–179.

[24] O. Oksuz, "Time-specific encrypted range query with minimum leakage disclosure," *IET Information Security*, vol. 15, no. 1, pp. 117–130, 2021.

[25] L. Baird, P. Mukherjee, and R. Sinha, "Temp: Time-locked encryption made practical." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 800, 2021.

[26] M. Ishizaka and S. Kiyomoto, "Time-specific encryption with constant-size secret-keys secure under standard assumption," *Cryptology ePrint Archive*, 2020.

[27] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007, pp. 350–364.

[28] M. Srivatsa, S. Balfe, K. G. Paterson, and P. Rohatgi, "Trust management for secure information flows," in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 175–188.

[29] K. Kasamatsu, T. Matsuda, K. Emura, N. Attrapadung, G. Hanaoka, and H. Imai, "Time-specific encryption from forward-secure encryption," in *Security and Cryptography for Networks: 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings 8*. Springer, 2012, pp. 184–204.

[30] R. Canetti, S. Halevi, and J. Katz, "A forward-secure public-key encryption scheme," in *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*. Springer, 2003, pp. 255–271.

[31] K. Kasamatsu, T. Matsuda, K. Emura, N. Attrapadung, G. Hanaoka, and H. Imai, "Time-specific encryption from forward-secure encryption: generic and direct constructions," *International Journal of Information Security*, vol. 15, pp. 549–571, 2016.

[32] K. Ogawa and K. Nuida, "Privacy preservation for versatile pay-tv services," in *HCI for Cybersecurity, Privacy and Trust: First International Conference, HCI-CPT 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings 21*. Springer, 2019, pp. 417–428.

[33] K. Ogawa, S. Tamura, and G. Hanaoka, "Key management for versatile pay-tv services," in *International Workshop on Security and Trust Management*. Springer, 2017, pp. 3–18.

[34] W. Wang, P. Xu, L. T. Yang, W. Susilo, and J. Chen, "Securely reinforcing synchronization for embedded online contests," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 2, pp. 1–21, 2017.

[35] J. Liu, T. Jager, S. A. Kakvi, and B. Warinschi, "How to build time-lock encryption," *Designs, Codes and Cryptography*, vol. 86, pp. 2549–2586, 2018.

[36] G. Choi and S. Vaudenay, "Timed-release encryption with master time bound key (full version)," *Cryptology ePrint Archive*, 2019.

[37] T. Zhang, H. Wu, and S. S. Chow, "Structure-preserving certificateless encryption and its application," in *Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings*. Springer, 2019, pp. 1–22.

[38] L. Zhang, "Research and practice of software protection based on virtual shell technology," in *Journal of Physics: Conference Series*, vol. 1656, no. 1. IOP Publishing, 2020, p. 012031.

[39] S. W. Lv, B. Su, P. Wang, Y. Y. Mao, and L. L. Huo, "Overview on sm4 algorithm," *Journal of Information Security Research*, vol. 2, no. 11, pp. 995–1007, 2016.

Ke Yuan is an Associate Professor in Henan University of China. He received the Ph.D. degree from Nankai University in 2014. His current interests include applied cryptography, cloud security, and artificial intelligence security.

Junyi Wu is a postgraduate student in Henan University of China. Her current interests include applied cryptography.

Baolei Zhang is a postgraduate student in Nankai University of China. His current interests are artificial intelligence and applied cryptography.

Bozhen Wang is an undergraduate student in Henan University of China. His current interest is applied cryptography.

Yuye Wang is an Associate Professor in Henan University of China. He received the Ph.D. degree from Harbin Engineering University in 2022. His current interests include privacy protection, social network analysis and applied cryptography.

Chunfu Jia is a Professor in Nankai University of China. He received the Ph.D. degree from Nankai University in 1996. His current interests include applied cryptography, computer system security, network security and artificial intelligence security.