Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Multiple time servers timed-release encryption based on Shamir secret sharing for EHR cloud system

Ke Yuan[1,2], Ziwei Cheng[1], Keyan Chen[1,3], Bozhen Wang[1], Junyang Sun[1], Sufang Zhou[1*] and Chunfu Jia[1,3]

Abstract
Electronic health record (EHR) cloud system, as a primary tool driving the informatization of medical data, have positively impacted both doctors and patients by providing accurate and complete patient information. However, ensuring the security of EHR cloud system remains a critical issue. Some patients require regular remote medical services, and controlling access to medical data involving patient privacy during specific times is essential. Timed-release encryption (TRE) technology enables the sender to preset a future time T at which the data can be decrypted and accessed. It is a cryptographic primitive with time-dependent properties. Currently, mainstream TRE schemes are based on non-interactive single time server methods. However, if the single time server is attacked or corrupted, it is easy to directly threaten the security applications of TRE. Although some research schemes "distribute" the single time server into multiple ones, they still cannot resist the single point of failure problem. To address this issue, we propose a multiple time servers TRE scheme based on Shamir secret sharing and another variant derived from it. In our proposed schemes, the data receiver does not need to interact with the time servers; instead, they only need to obtain the time trapdoors that exceed or equal the preset threshold value for decryption, which ensures the identity privacy of the data sender and tolerates partial downtime or other failures of some time servers, significantly improving TRE reliability. Security analysis indicates that our proposed schemes demonstrate data confidentiality, verifiability, anti-advance decryption, and robust decryption with multiple time trapdoors, making them more practical. Efficiency analysis indicates that although our schemes have slightly higher computational costs than most efficient existing TRE schemes, such differences are insignificant from a practical application perspective.

Keywords Timed-release encryption, Multiple time servers, Shamir secret sharing, Provable security, Electronic health record

Introduction

With the advent of the information age, healthcare institutions are rapidly evolving towards informatization, giving rise to electronic health record (EHR) cloud system [1]. EHR cloud system significantly enhances productivity in resource sharing, providing robust support for healthcare professionals. Including comprehensive patient information, EHR cloud system enables medical teams to have a more holistic understanding of patients' medical history, facilitating in-depth assessments and faster diagnoses. By digitizing and centrally managing patient medical information, healthcare personnel can easily access necessary data to support decision-making and the execution of medical plans [2, 3].

Cloud computing, a computing paradigm based on the internet, plays a crucial role in healthcare data

*Correspondence:
Sufang Zhou
zsf@henu.edu.cn
[1] School of Computer and Information Engineering, Henan University, Kaifeng 475004, Henan, China
[2] Henan Province Engineering Research Center of Spatial Information Processing, Henan University, Kaifeng 475004, Henan, China
[3] College of Cybersecurity, Nankai University, Tianjin 300350, China

© The Author(s) 2024. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

management by providing secure and reliable solutions for storing and processing large-scale medical data [4–6]. Cloud computing facilitates rapid access, sharing, and analysis of medical data, offering comprehensive support for healthcare decision-making. Additionally, the elastic and automated features of cloud computing enable healthcare institutions to adjust resources according to needs, improving data management efficiency and fostering innovation in medical research and patient care.

Despite the flexibility and efficiency brought by cloud computing to healthcare data management, security remains a critical concern. Particularly in the handling of patient privacy data, cloud storage, and access services may pose risks of data leakage, leading to the unauthorized disclosure of sensitive patient information [5]. For example, in the case of patients with chronic diseases like diabetes, who regularly upload data through remote monitoring devices, there is a potential for unauthorized data access if this physiological data is stored on a EHR cloud system. In such scenarios, a cryptographic technology that can control the decryption time becomes a key technology to ensure patient privacy. Timed-release encryption (TRE) allows users to preset decryption time, and access is only permitted after the decryption time, effectively preventing unauthorized privacy infringements. For instance, a medical cloud system could use a multiple time servers scheme to encrypt the physiological data of each patient and set a specific decryption time. At the designated weekly decryption time, doctors can decrypt and analyze the patient's physiological data for regular remote assessments. This periodic assessment helps doctors better understand the patient's health condition. Such a security measure not only provides more reliable privacy protection for patients but also ensures the security of sensitive medical data on the EHR cloud system.

The setting of specific decryption time is not just for security; it is based on a series of reasonable considerations. Firstly, it helps prevent patients from excessive anxiety, as they know that doctors will only review the data in the specific time, allowing them to focus on daily life during this period and alleviate unnecessary worries. Secondly, this method encourages patients to actively participate in their health management, showcasing better physiological data. Moreover, it avoids premature intervention in medical decisions, ensuring that doctors make accurate medical decisions with sufficiently stable data. Lastly, this security measure simultaneously upholds patient privacy rights by limiting access to data, reducing the risk of data misuse or improper use, and providing more reliable privacy protection for patients. This periodic assessment not only helps doctors better understand the patient's health condition but also ensures the security of sensitive medical data on the EHR cloud system while safeguarding patient privacy.

Therefore, TRE with specific decryption times is crucial in medical practice, not only ensuring security but also promoting the patient recovery process, becoming an important and meaningful component of medical data management. This paper aims to propose a multiple time servers TRE scheme based on Shamir secret sharing for EHR cloud system. The data receiver only needs to obtain time trapdoors published by time servers exceeding or equal to the threshold value. This ensures that the decryption process can be completed even in the event of time server failures or other faults, enhancing the system's fault tolerance and the reliability of data decryption.

Related work

TRE [7, 8] is a cryptographic primitive that can control the decryption time. Its core idea is to introduce the time factor into the general encryption scheme so that the receiver can only decrypt the ciphertext at a specified time in the future. TRE is suitable for solving many time-dependent real-world and virtual applications, such as sealed bidding, timed release of electronic documents, and electronic voting blockchain applications [9], etc.

The TRE technology was first proposed in May [7]. In 1996, Rivest et al. [10] proposed two foundational TRE construction schemes: one based on time-lock puzzles (TLP) that relies on the factorization problem and another involving sender-proxy interactions for time and message release. These laid the theoretical foundation for sustained research in the field of TRE. Currently, TRE construction schemes include TLP methods [11–16], proxy methods [17–22], and other methods [23–29]. In the TLP-based TRE schemes, the decryption key is hidden in a mathematical formula. After the sender sends the ciphertext, the receiver needs to perform a large number of calculations. Among the TRE schemes based on other methods (network methods, quantum methods), for example, Unruh et al. [27] achieved revocable TRE based on quantum cryptography without trusted parties. Li et al. [26] explored a timed-release data scheme based on the blockchain network's smart contracts, recruiting several network nodes as middlemen (each middleman needs to pay a deposit) to send decryption keys to receivers at specified decryption time T. Chae et al. [28] proposed a timed-release blockchain scheme that combines blockchain PoW algorithms with TLP algorithms. Compared with schemes proposed by Liu et al. [25] and Malavolta et al. [29], it employs standard encryption without requiring additional computational work, and its feasibility has been evaluated in an electronic voting application system.

Currently, most TRE schemes are constructed based on the time server approach. Depending on whether the receiver needs to interact with the time server, they can be divided into interactive and non-interactive time server modes. The former requires users to interact with the time server, which cannot guarantee user anonymity and may easily lead to denial-of-service attacks causing system paralysis, thus limiting the scalability of the scheme. In contrast, in TRE schemes constructed using the latter approach, the time server does not need to interact with users and may even be unaware of their existence. It only needs to calculate and broadcast a short signature-formatted time trapdoor at a specified time, ensuring the anonymity of user information and better scalability. Researchers have attempted to construct multiple time servers TRE schemes to prevent single-point attacks or corruption by attackers to reduce the risk of attackers breaking the entire TRE model. In 2021, Yuan et al. [30] proposed a non-interactive multiple time servers TRE scheme (MTSTRE scheme), which is the most efficient multiple time servers scheme. However, if one of the time servers fails, the data receiver will fail to decrypt the data normally at the specified time T. Therefore, this scheme has some defects in practicability.

Secret sharing techniques [31–34] can split a secret into multiple secret shares, allowing partial secret shares to reconstruct the complete secret. By appropriately utilizing this technology, this paper integrates the Shamir secret sharing technique into the MTSTRE scheme and designs a non-interactive TRE model for multiple time servers based on secret sharing (SS-MSTRE). This model allows for partial time trapdoor failure while still enabling data receivers to decrypt promptly, thus improving practicality.

Our contributions

We address the issue of the single point of failure problem in plain multiple time servers TRE schemes and propose a more practical SS-MSTRE scheme. Our main contributions are as follows:

- We migrate the Shamir secret sharing technique from prime fields to elliptic curve groups, enabling its use in cryptographic scheme constructions based on bilinear pairing-related hard problems.
- We integrate Shamir secret sharing over elliptic curve groups into the construction of multiple time servers TRE cryptographic schemes, designing a more practical SS-MSTRE model and constructing a provably secure concrete scheme and its variants. When the specified decryption time arrives, even if some time trapdoors fail, the data receiver can still decrypt the ciphertext on time using time trapdoors exceeding or equal to the threshold. In addition, it increases the cost of attacking or bribing the time server to decrypt the data received by the receiver or attacker in advance.
- We employ identity-based encryption (IBE) technology to encrypt key shares to ensure secure and highly efficient distribution and transmission of key shares.
- In real-world scenarios, there may be situations where the time server management organization is not trusted. If the private key of the time server management organization is compromised, it could lead to obtaining the master time trapdoor, allowing for premature decryption of ciphertexts. Therefore, the key shares provided directly by the time server management organization cannot be used as the time server's private key. So, we further propose the SS-MSTRE$_2$ scheme. In the SS-MSTRE$_2$ scheme, the time server's private key is jointly generated by the time server management organization and a random number, thus enhancing the security of the scheme.

Preliminary

In this section, we present the key notations involved in our schemes and briefly review the basic content of bilinear pairing, bilinear Diffie-Hellman (BDH) assumption, Shamir secret sharing algorithm, and the identity-based encryption scheme.

Key notations

For the convenience of understanding, we have given the key notations used in our schemes in Table 1.

Bilinear pairing

We give a form of bilinear pair and its properties, as follows.

Definition 1 Suppose G_1 is an elliptic curve discrete logarithmic problem(ECDLP) additive group over a finite field, G_2 is a discrete logarithmic problem(DLP) multiplicative group over a finite field, and the order of G_1 and G_2 is a prime number q. Using the bilinear pairing technique, the ECDLP additive group over a finite field can be reduced to the DLP multiplicative group over a finite field. The bilinear map is $e : G_1 \times G_1 \longrightarrow G_2$, satisfying the following properties:

(1) Bilinear. For any $P, Q, R \in G_1$, there are

$$\begin{aligned} e(P+Q,R) &= e(P,R)e(Q,R) \\ e(P,Q+R) &= e(P,Q)e(P,R) \end{aligned} \tag{1}$$

Table 1 Key notations

Symbol	Description
EHR	Electronic health record
TRE	Timed-release encryption
T	Preset decryption time by the data sender
t	The preset threshold value
SS-MSTRE	A non-interactive TRE model for multiple time servers based on secret sharing
IBE	Identity-based encryption
C	The ciphertext
M	The plaintext
k	The security parameter during system initialization
λ	The security parameter during private key generator initialization
sk	The time server management organization's private key
$params_{tsmo}$	The time server management organization's system parameters
$p, \mathbb{p}$	Prime orders
$G_1, \mathbb{G}_1$	ECDLP additive groups
$G_2, \mathbb{G}_2$	DLP multiplicative groups
$P, \mathbb{P}$	Random generators
$e, \mathbb{e}$	Bilinear mappings
n	The length of the message
$H_1, H_2, \mathbb{H}_1, \mathbb{H}_2, \mathbb{H}_3, \mathbb{H}_4$	Secure hash functions
MSK	The private key generator's secret key
MPK	The private key generator's public key
$params_{pkg}$	The private key generator's system parameters
IDs	N time servers's identity identifiers
TS_i	The ith time server
$temp_{priv}^{(i)}$	The time server TS_i's temporary private key
$temp_{pub}^{(i)}$	The time server TS_i's temporary public key
$ts_{priv}^{(i)}$	The time server TS_i's private key
$ts_{pub}^{(i)}$	The time server TS_i's public key
usk	The data receiver's private key
upk	The data receiver's public key
$t_{instance}$	Time instance
$S_T^{(i)}$	The time trapdoor generated by the ith time server TS_i
U_T	The time trapdoor calculated by the data receiver
Xs	The corresponding set of identification numbers for the time servers

(2) Nondegeneracy. If g is a generator of G_1, then $e(g, g)$ is a generator of G_2.
(3) Computability. For any $P, Q \in G_1$, there is an effective algorithm to calculate $e(P, Q)$.

From the above properties, we can further deduce the property that the coefficients of bilinear pair elements can move freely, that is, $e(aP, bQ) = e(abP, Q) = e(p, abQ) = e(bP, aQ) = e(P, Q)^{ab}$. Admissible bilinear pairings can be constructed via the Weil and Tate pairings [35, 36].

BDH assumption

The bilinear Diffie-Hellman (BDH) assumption plays a crucial role in the design of TRE schemes.

Definition 2 Given $P, aP, bP, cP \in G_1$, where $a, b, c \in \mathbb{Z}_p^*$ are unknown, the goal is to calculate $e(P, P)^{abc}$, where e is a bilinear mapping and P is a generator of G_1 as defined in Definition 1.

If $\Pr[\mathcal{A}(P, aP, bP, cP) = e(P, P)^{abc}] \geq \varepsilon$, then the advantage of the adversary $\mathcal{A}$ to overcome the BDH assumption is ε, and ε is negligible.

Shamir secret sharing

Our schemes use the Shamir secret sharing algorithm to deal with the failure of partial time trapdoors when the specified decryption time comes. In the following, we give the basic flow of Shamir secret sharing algorithm and the definition of its access structure.

(1) *Protocol initialization algorithm.* The distributor of confidential information randomly selects n different non-zero elements $x_1, x_2, x_3, ..., x_n$ from the finite field $GF(p)$ as the unique identification numbers corresponding to n participants P_i $(i = 1, 2, ..., n)$, p is prime and $p \gg n$.
(2) *Secret distribution algorithm.* The distributor selects the secrets to be distributed, randomly selects $(t-1)$ elements $a_1, a_2, ..., a_{t-1}$ from the finite field $GF(p)$, and constructs the secret sharing polynomial $f(x) = \left(s + \sum_{i=1}^{t-1} a_i x^i\right) \bmod p$, calculates $s_i = f(x_i)$ and sends it to the corresponding participant p_i $(i = 1, 2, ..., n)$ as a secret share.
(3) *Secret reconstruction algorithm.* If any t of n participants shows their secret shares $(x_1, s_1), (x_2, s_2), \cdots, (x_t, s_t)$, the Lagrange interpolation polynomial can be reconstructed as follows:

$$L(x) = s_1 \frac{(x-x_2)(x-x_3)\cdots(x-x_t)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_t)} + s_2 \frac{(x-x_1)(x-x_3)\cdots(x-x_t)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_t)} + \cdots + s_t \frac{(x-x_1)(x-x_2)\cdots(x-x_{t-1})}{(x_t-x_1)(x_t-x_2)\cdots(x_t-x_{t-1})} = \sum_{i=1}^{t} s_i \delta_i(x) \bmod p \quad (2)$$

where

$$\delta_i(x) = \prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \quad (3)$$

and the secret $s = L(0)$ can be calculated.

Definition 3 Access structure. Suppose the set of n participants is $P = \{p_i | i = 1, 2, ..., n\}$ and Γ is an access structure on set P, where $\Gamma \subseteq 2^{|p|}$, $2^{|p|}$ represents all sets of non-empty subsets on set P, satisfying the following properties.

① If $A \in \Gamma$, $A \subseteq B \subseteq \Gamma$, then $B \in \Gamma$.
② Participants in set Γ can reconstruct the secret.

Identity-based encryption

We use identity-based encryption (IBE) technology to ensure the security of key shares during transmission between different devices and to provide verifiable attributes for these key shares. Compared to other encryption methods, IBE significantly simplifies key management operations. Its advantages include the elimination of the need to associate public keys with extensive public key infrastructure (PKI), no requirement for digital certificates, no reliance on online certificate authority (CA), reduced key lengths, and enhanced security.

Definition 4 ξ_{IBE}={IBE.Setup, IBE.Extract, IBE.Encrypt, IBE.Decrypt}. The IBE.Encrypt algorithm and the IBE. The decrypt algorithm satisfies the consistency constraint. Namely, given any plaintext M, ciphertext C can be obtained by IBE.Encrypt algorithm, and we can also decrypt and recover plaintext M by IBE.Decrypt algorithm.

IBE.Setup(1^k). Given a security parameter 1^k, this algorithm outputs public parameters PP and the master secret key mk.

IBE.Extract (PP, mk, ID). Given a unique identifier $ID \in \{0,1\}^*$ that can distinguish user identity information, the master key mk, and the public parameters PP, this algorithm outputs the corresponding private key d_{ID}.

IBE.Encrypt (PP, ID, M). Given plaintext M, public parameters PP, and an identifier ID that can distinguish user identity information, this algorithm outputs the corresponding ciphertext C.

IBE.Decrypt(PP, C, d_{ID}). Given ciphertext C, public parameters PP, and the user's private key d_{ID}, this algorithm outputs the corresponding plaintext M.

System and security model

System model

The design goal of the proposed schemes is that the receiver can decrypt the ciphertext C normally at the decryption time specified by the sender. In this section, we introduce a common time server management organization to the system and further present our TRE system model based on Shamir secret sharing, as shown in Fig. 1. The system consists of five entities: the time server management organization, N time servers, the private key generator, the data sender, and the data receiver.

Time server management organization. The time server management organization is a fully trusted entity in the SS-MSTRE$_1$ scheme, while it is a semi-trusted entity in the SS-MSTRE$_2$ scheme. It is responsible for generating system parameters to initialize the system and using the Shamir secret sharing algorithm to generate key shares of N time servers. Simultaneously, utilizing the *IBE.Encrypt* algorithm defined in Definition 4, sends the key shares to the corresponding time servers as their respective private keys.

***N* time servers.** N time servers are semi-trusted entities responsible for providing an accurate time reference to the data receiver. In the proposed schemes, there is no need for interaction between N time servers and the data receiver, and they are responsible for broadcasting time trapdoors at a fixed frequency, such as every five minutes.

Private key generator. The private key generator is trusted for all N time servers. It is responsible for correctly executing each calculational task for every time server, including using the *IBE.Extract* the algorithm defined in Definition 4 to generate temporary public-private key pairs for N time servers. These temporary keys are used for data transmission between the time servers and the time server management organization.

Data sender. The data sender is a user who wishes the encrypted data to be decrypted at a specified time and is responsible for specifying the decryption time T,

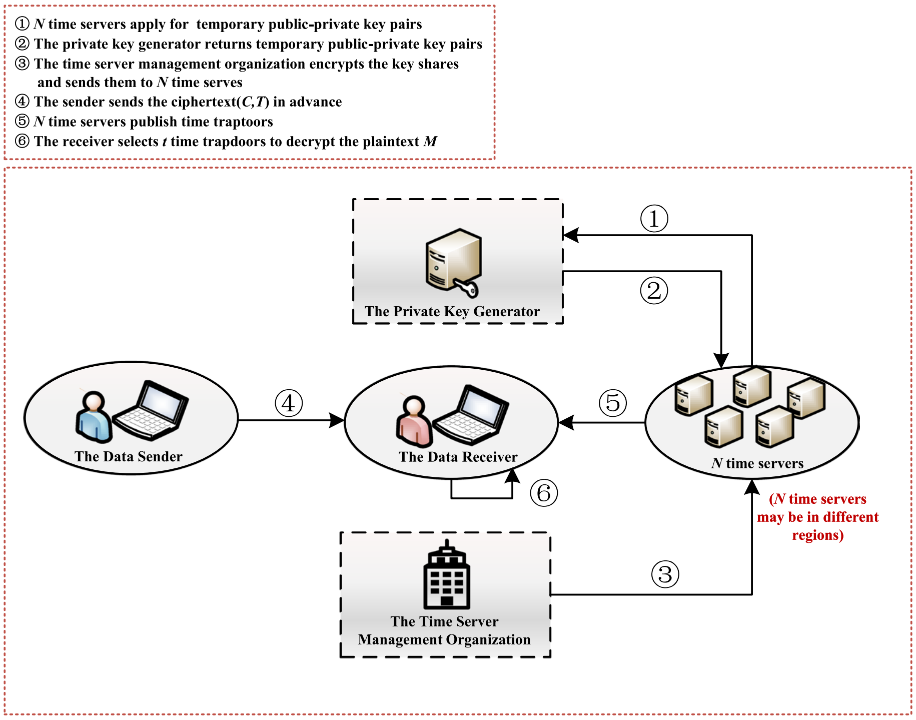
① N time servers apply for temporary public-private key pairs
② The private key generator returns temporary public-private key pairs
③ The time server management organization encrypts the key shares and sends them to N time serves
④ The sender sends the ciphertext(C,T) in advance
⑤ N time servers publish time traptoors
⑥ The receiver selects t time trapdoors to decrypt the plaintext M
The Private Key Generator
The Data Sender
The Data Receiver
N time servers
(N time servers may be in different regions)
The Time Server Management Organization

Fig. 1 SS-MSTRE system

encrypting the plaintext M, and sending the ciphertext (C, T) to the data receiver.

Data receiver. The data receiver is a user who can only decrypt C at a specified time T by the data sender. To complete decryption, they must select at least t valid time trapdoors from multiple servers' published time trapdoors.

Security model

In this paper, we make the following assumptions:

(1) The private key generator is entirely trustworthy and can accurately perform computational tasks for each time server.
(2) The system has sufficient time servers operating normally to ensure that decryption can proceed normally.
(3) N time servers are honest but curious, meaning that they will follow the rules for providing services. However, they may save the input and output results to infer information related to decrypting the ciphertext sent by the sender.

The proposed schemes possess data confidentiality, verifiability, anti-advance decryption, and robust decryption with multiple time trapdoors. We will provide a detailed analysis in Security analysis section.

(1) Data confidentiality. It should be ensured that attackers cannot illegally analyze the key information required for decrypting the ciphertext before the specified decryption time T.
(2) Verifiability. It should use some algorithms or methods to verify the validity and correctness of intermediate data to detect any tampering with the intermediate data.
(3) Anti-advance decryption. It should prevent dishonest receivers from decrypting EHR before the specified decryption time.
(4) Robust decryption with multiple time trapdoors. It should be ensured that even if some time servers fail or are attacked, the data receiver can still use other sufficient time trapdoors for decryption.

Algorithm definition

Definition 5 Our non-interactive SS-MSTRE system includes five entities: the time server management organization, N time servers, the private key generator PKG, the data sender, the data receiver, and algorithm 10-tuple $\mathcal{E}_{\mathrm{SS-MSTRE}} = \{TSMO\_Setup, PKG\_Setup, TempKey\_Extract, KeySharing, TS\_KeyGen, User\_KeyGen, Enc, TS\_Rel, US\_Rel, Dec\}$.

$TSMO\_Setup(k)$. It is a probabilistic initialization algorithm. Given a security parameter k, this algorithm outputs the private key sk of the time server management organization and the system parameters $params_{\mathrm{tsmo}}$.

$PKG\_Setup(\lambda)$. Given a security parameter λ, this algorithm outputs the master secret key MSK, the public key MPK, and the system parameters $params_{\text{pkg}}$ of the private key generator.

$TempKey\_Extract(IDs, params_{\text{pkg}}, MSK)$. Given a set of identity identifiers for N time servers IDs, the private key generator's system parameters $params_{\text{pkg}}$, and the private key generator's secret key MSK, this algorithm outputs a temporary public-private key pairs set $temp$ for N time servers.

$KeySharing(sk, temp, MPK, params_{\text{pkg}})$. Given the public key sk of the time server management organization, a temporary public-private key pairs set $temp$ of N time servers, the public key MPK of the private key generator, and the system parameters $params_{\text{pkg}}$, this algorithm outputs N key share ciphertexts $\{C_1, C_2, \ldots, C_N\}$.

$TS\_KeyGen\left(C_i, temp_{priv}^{(i)}, params_{\text{pkg}}, params_{\text{tsmo}}\right)$. Given the key share ciphertext C_i corresponding to the time server TS_i, the temporary private key $temp_{priv}^{(i)}$ of the time server TS_i, the private key generator's system parameters $params_{\text{pkg}}$, and system parameters $params_{\text{tsmo}}$, this algorithm outputs the public key $ts_{pub}^{(i)}$ and private key $ts_{priv}^{(i)}$ of the time server TS_i.

$User\_KeyGen(params_{\text{tsmo}})$. This is a probabilistic algorithm for key generation. Given the system parameters $params_{\text{tsmo}}$, this algorithm outputs the data receiver's private key usk and public key upk.

$Enc\left(M, upk, ts_{pub}^{(i)}(i = 1, 2, ..., N), T, params_{\text{tsmo}}\right)$. This is a probabilistic encryption algorithm. Given an EHR record M, receiver's public key upk, time servers' public key $ts_{pub}^{(i)}(i = 1, 2, ..., N)$, decryption time T specified by the data sender, and system parameters $params_{\text{tsmo}}$, this algorithm outputs the ciphertext C.

$TS\_Rel\left(ts_{priv}^{(i)}, t_{instance}, params_{\text{tsmo}}\right)$. This is a probabilistic algorithm for generating time trapdoors. Given the time server TS_i's private key $ts_{priv}^{(i)}$, time instance $t_{instance}$, and system parameters $params_{\text{tsmo}}$, this algorithm outputs the corresponding time trapdoor $S_T^{(i)}$.

$US\_Rel(usk, T, params_{\text{tsmo}})$. This is a probabilistic algorithm for generating the user's time trapdoor. Given the data receiver's private key usk, specified decryption time T, and system parameters $params_{\text{tsmo}}$, this algorithm outputs the time trapdoor U_T of the data receiver.

$Dec(C, STs, Xs, T, U_T, params_{\text{tsmo}})$. This is a deterministic algorithm for joint decryption. Given the ciphertext C, the set of effective time trapdoors STs chosen by the data receiver, the set of identification numbers Xs corresponding to time servers, specified decryption time T, the data receiver's time trapdoor U_T, and system parameters $params_{\text{tsmo}}$, this algorithm outputs the plaintext M or $\perp$.

Concrete schemes of SS-MSTRE

This section constructs two concrete SS-MSTRE schemes based on whether the time server management organization is trusted: SS-MSTRE$_1$ and SS-MSTRE$_2$. In SS-MSTRE$_1$, we assume that the time server management organization is trusted. In SS-MSTRE$_2$, we assume that the time server management organization is semi-trusted.

Construction of SS-MSTRE$_1$

Our non-interactive SS-MSTRE$_1$ works as follows:

(1) $(params_{\text{tsmo}}, sk) \leftarrow TSMO\_Setup(k)$. The time server management organization runs the $TSMO\_Setup$ algorithm to generate the system initialization parameters. The time server management organization selects the security parameter k and performs the following operations:

① Selects a prime order p, G_1 and G_2 are a p-order ECDLP additive group and DLP multiplicative group respectively.
② Selects a random generator $P \in G_1$.
③ Selects a bilinear mapping $e : G_1 \times G_1 \rightarrow G_2$ satisfies Definition 1.
④ Select two secure hash functions: H_1:$\{0,1\}^* \rightarrow G_1$ and H_2:$G_2 \rightarrow \{0,1\}^n$, where n represents the length of the message.
⑤ Selects a random number $s \in \mathbb{Z}_p^*$ as its private key $sk = s \in \mathbb{Z}_p^*$.
⑥ Defines a threshold value t.
⑦ Outputs the system parameters $params_{\text{tsmo}}$= $\{p, P, G_1, G_2, e, H_1, H_2, n, t\}$ and the private key sk.

(2) $(params_{\text{pkg}}, MPK, MSK) \leftarrow PKG\_Setup(\lambda)$. The private key generator runs the PKG_Setup algorithm to generate its initialization parameters. The private key generator selects the security parameter λ and performs the following operations:

① Selects a prime order $\mathbb{p}$, $\mathbb{G}_1$ and $\mathbb{G}_2$ are a $\mathbb{p}$-order ECDLP additive group and DLP multiplicative group respectively.
② Selects a random generator $\mathbb{P} \in \mathbb{G}_1$.
③ Selects a bilinear mapping $\mathbb{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies Definition 1.
④ Selects four secure hash functions: $\mathbb{H}_1$:$\{0,1\}^* \rightarrow \mathbb{G}_1$, $\mathbb{H}_2$:$\mathbb{G}_2 \rightarrow \{0,1\}^n$, $\mathbb{H}_3$:$\{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_{\mathbb{p}}^*$, $\mathbb{H}_4$:$\{0,1\}^n \rightarrow \{0,1\}^n$.
⑤ Selects a random number $\mathbb{a} \in \mathbb{Z}_{\mathbb{p}}^*$ as its master secret key $MSK = \mathbb{a} \in \mathbb{Z}_{\mathbb{p}}^*$ and calculates its master public key $MPK = \mathbb{a}\mathbb{P}$.

⑥ Outputs the private key generator's system parameters $params_{\text{pkg}} = \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{n}, \mathbb{p}, \mathbb{e}, \mathbb{P}, \mathbb{H}_1, \mathbb{H}_2, \mathbb{H}_3, \mathbb{H}_4, MPK\}$ and the master secret key MSK.

(3) $temp \leftarrow TempKey\_Extract(IDs, params_{\text{pkg}}, MSK)$. The private key generator runs the *TempKey_Extract* algorithm to generate a set of temporary public-private key pairs for N time servers. The following steps are required:
① Calculates the set of temporary public-private key pairs $temp = \left\{temp_{pub}^{(i)}, temp_{priv}^{(i)}\right\}_{i=1,2,..,N}$ for N time servers using their identity identifier set $IDs = \{ID_1, ID_2, ..., ID_N\}$, where $temp_{\text{priv}}^{(i)} = MSK \cdot \mathbb{H}_1(ID_i) = \mathbb{a}\mathbb{H}_1(ID_i)$, $temp_{\text{pub}}^{(i)} = \mathbb{H}_1(ID_i)$, $ID_i \in IDs$.

(4) $(C_1, C_2, ..., C_N) \leftarrow KeySharing(sk, temp, MPK, params_{\text{pkg}})$. The time server management organization runs the *KeySharing* algorithm to generate secret key shares for N time servers. The following steps are required:

① The time server management organization selects secret sharing polynomial coefficients $a_1, a_2, a_3, \ldots, a_{t-1} \in \mathbb{Z}_p^*$ to construct Shamir secret sharing polynomial $f(x) = \left(s + \sum_{i=1}^{t-1} a_i x^i\right) \bmod p$, generate secret key share $s_i = f(x_i)$ for time server TS_i $(i = 1, 2, ..., N)$, where $x_i = i$ is the identification number of time server TS_i.
② The time server management organization selects a random number $\sigma \in \{0,1\}^n$ and uses the *IBE.Encrypt* algorithm defined in Definition 4 to get the ciphertext C_i, which is sent to time server TS_i using the temporary public key $temp_{pub}^{(i)}$, where $i = 1, 2, ..., N$.

$$C_i =< U, V, W >=< \mathbb{r}\mathbb{P}, \sigma \oplus \mathbb{H}_2\left(\mathbb{e}\left(MPK, temp_{pub}^{(i)}\right)^{\mathbb{r}}\right), s_i \oplus \mathbb{H}_4(\sigma) >, \quad \mathbb{r} = \mathbb{H}_3(\sigma, s_i) \tag{4}$$

(5) $\left(ts_{pub}^{(i)}, ts_{priv}^{(i)}\right) \leftarrow TS\_KeyGen\left(C_i, temp_{priv}^{(i)}, params_{\text{pkg}}, params_{\text{tsmo}}\right)$. The time server TS_i runs the *TS_KeyGen* algorithm to obtain its public key $ts_{pub}^{(i)}$ and private key $ts_{priv}^{(i)}$. The following steps are required:
① The time server $TS_i (i = 1, 2, ..., N)$ receives the ciphertext C_i from the time server management organization. It decrypts the ciphertext C_i using its temporary private key $temp_{priv}^{(i)}$ to obtain its private key $ts_{priv}^{(i)}$. The time server TS_i then calculates its public key $ts_{pub}^{(i)}$.

$$\begin{aligned} ts_{priv}^{(i)} &= W \oplus \mathbb{H}_4(\sigma) = s_i, \sigma = V \oplus \mathbb{H}_2(e(temp_{priv}^{(i)}, U)) \\ ts_{pub}^{(i)} &= ts_{priv}^{(i)} P = s_i P \end{aligned} \tag{5}$$

(6) $(usk, upk) \leftarrow User\_KeyGen(params_{\text{tsmo}})$. The data receiver runs the *User_KeyGen* algorithm to obtain its private key usk and its public key upk. The following steps are required:
① The data receiver selects a random number $u \in \mathbb{Z}_p^*$ as its private key $usk = u$, and calculates its public key $upk = uP$.

(7) $C \leftarrow Enc\left(M, upk, ts_{pub}^{(i)}(i = 1, 2, ..., N), T, params_{\text{tsmo}}\right)$. The data sender runs the *Enc* algorithm using the data receiver's public key upk, and arbitrarily selects at least t public keys from N time servers' public keys $ts_{pub}^{(i)}$ to form a set ts_{pub}. Xs is the corresponding set of identification numbers for the time servers. The data sender specifies a decryption time $T \in \{0,1\}^*$ to encrypt M. The following steps are required:

① Calculates

$$tk = \sum_{ts_{\text{pub}}^{(i)} \in \text{ts}_{\text{pub}}} ts_{\text{pub}}^{(i)} \prod_{x_j \in X_s, x_j \neq x_i} \frac{-x_j}{(x_i - x_j)} = sP \tag{6}$$

② Selects a random number $r \in \mathbb{Z}_p^*$ and calculates $K = e(rH_1(T), upk + tk)$.
③ Outputs the ciphertext $C =< X = rP, Y = M \oplus H_2(K) >$.

(8) $S_T^{(i)} \leftarrow TS\_Rel\left(ts_{priv}^{(i)}, t_{instance}\right)$. The time server runs the *TS_Rel* algorithm at a fixed frequency (for example, every five minutes) to broadcast the time trapdoor. The following steps are required:
① On the time instance $t_{instance} \in \{0,1\}^*$, the time server TS_i calculates and periodically broadcasts the time trapdoor

$$S_T^{(i)} = ts_{priv}^{(i)} H_1(t_{instance}) = s_i H_1(t_{instance}) \tag{7}$$

to all system users using its private key.

(9) $U_T \leftarrow US\_Rel(usk, T, params_{\text{tsmo}})$. The data receiver uses the decryption time T specified by the data sender and their private key usk to run the *US_Rel* algorithm and obtain their time trapdoor. The following steps are required:
① At the specified decryption time T, the data receiver calculates the time trapdoor $U_T = usk \cdot H_1(T) = uH_1(T)$ using the private key usk.

(10) $M \leftarrow Dec(C, STs, Xs, T, U_T, params_{\text{tsmo}})$. The data receiver runs the *Dec* algorithm to recover the plaintext M. The following steps are required:

① At the decryption time T specified by the data sender, each of the N time servers sends the corresponding time trapdoor ST so that there are N time trapdoors STs. The data receiver randomly selects a set of valid time trapdoors STs from the time trapdoors published by N time servers, ensuring that $|STs| \geq t$. Xs is the corresponding set of identification numbers for the time servers. Calculate the main time trapdoor $S'_T = \sum_{S_T^{(i)} \in STs} S_T^{(i)} \prod_{x_j \in Xs, x_j \neq x_i} \frac{-x_j}{(x_i - x_j)}$.
② Calculates $K' = e(X, S'_T + U_T)$.
③ Recovers the plaintext $M = Y \oplus H_2(K')$.

Assume that the ciphertext is $C = \langle X, Y \rangle$, the decryption time is T, the set of valid time trapdoors is $STs = \{S_T^{(1)}, S_T^{(2)}, \ldots, S_T^{(t)}\}$, the corresponding set of identification numbers for the time servers is $Xs = \{x_1, x_2, \ldots, x_t\}$, and the user's trapdoor is U_T. The correctness of decryption is verified as follows:

$$\begin{aligned}
S'_T &= \sum_{S_T^{(i)} \in STs} S_T^{(i)} \prod_{x_j \in Xs, x_j \neq x_i} \frac{-x_j}{x_i - x_j} \\
&= s_1 H_1(T) \cdot \frac{(0-x_2)(0-x_3)\cdots(0-x_t)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_t)} \\
&\quad + s_2 H_1(T) \cdot \frac{(0-x_1)(0-x_3)\cdots(0-x_t)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_t)} + \cdots \\
&\quad + s_t H_1(T) \cdot \frac{(0-x_1)(0-x_2)\cdots(0-x_{t-1})}{(x_t-x_1)(x_t-x_2)\cdots(x_t-x_{t-1})} \\
&= \Bigg(s_1 \cdot \frac{(0-x_2)(0-x_3)\cdots(0-x_t)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_t)} + \\
&\quad s_2 \cdot \frac{(0-x_1)(0-x_3)\cdots(0-x_t)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_t)} + \cdots \\
&\quad + s_t \cdot \frac{(0-x_1)(0-x_2)\cdots(0-x_{t-1})}{(x_t-x_1)(x_t-x_2)\cdots(x_t-x_{t-1})} \Bigg) \cdot H_1(T) \\
&= sH_1(T) \\
K' &= e(X, S'_T + U_T) \\
&= e(rP, sH_1(T) + uH_1(T)) \\
&= e(rH_1(T), upk + tk) \\
&= K \\
Y \oplus H_2(K') &= Y \oplus H_2(K) \\
&= M \oplus H_2(K) \oplus H_2(K) \\
&= M
\end{aligned} \tag{8}$$

Construction of SS-MSTRE$_2$

In the real world, the time server management organization may be semi-trusted, so it is not possible to directly use the key shares published by the time server management organization as private keys for time servers. To solve this problem, it is necessary to improve the SS-MSTRE$_1$ scheme. The difference between the improved scheme and the SS-MSTRE$_1$ scheme is that each time server, after decrypting the key share ciphertext obtained using the *IBE.Decrypt* algorithm defined in Definition 4, does not directly use the obtained key share itself as its private key. Instead, N time servers first "negotiate" a shared random number. Each time server then uses this shared random number and the decrypted key share to generate a new private key. The specific improvement method is as follows:

① At system initialization (ensuring that N time servers are all in normal working state), N time servers need to specify a particular time server TS_j to generate a random number $R \in \mathbb{Z}_p^*$. Then, using the *IBE.Encrypt* algorithm defined in Definition 4, the time server TS_j sends the random number R to the other time servers TS_i , where $i \neq j$. The time server TS_i then uses the *IBE.Decrypt* algorithm defined in Definition 4 to obtain the shared random number R.
② The time server runs the *TS_KeyGen* algorithm to obtain s_i and uses the shared random number R to calculate its private key $ts_{priv}^{(i)} = s_i R$, then calculates its public key $ts_{pub}^{(i)} = ts_{priv}^{(i)} P = s_i RP$.

Correspondingly, the *Enc* algorithm is modified as follows:
$C \leftarrow Enc(M, upk, ts_{pub}^{(i)}(i = 1, 2, ..., N), T, params_{\text{tsmo}})$. The data sender runs the *Enc* algorithm using the data receiver's public key *upk*, and arbitrarily selects at least t public keys from N time servers' public keys $ts_{pub}^{(i)}$ to form a set ts_{pub}. Xs is the corresponding set of identification numbers for the time servers. The data sender specifies a decryption time $T \in \{0,1\}^*$ to encrypt the data M. The following steps are required:

① Calculates

$$tk = \sum_{ts_{pub}^{(i)} \in ts_{pub}} ts_{pub}^{(i)} \prod_{x_j \in X_s, x_j \neq x_i} \frac{-x_j}{(x_i - x_j)} = sRP. \tag{9}$$

② Selects a random number $r \in \mathbb{Z}_p^*$ and calculates $K = e(rH_1(T), upk + tk)$.
③ Outputs the ciphertext $C =< X = rP, Y = M \oplus H_2(K) >$.

Correspondingly, the *TS_Rel* algorithm is modified as follows:
$S_T^{(i)} \leftarrow TS\_Rel(ts_{priv}^{(i)}, t_{instance})$. The time server runs the *TS_Rel* algorithm at a fixed frequency (for example, every five minutes) to broadcast the time trapdoor. The following steps are required:

① On the time instance $t_{instance} \in \{0,1\}^*$, the time server TS_i calculates and periodically broadcasts the time trapdoor $S_T^{(i)} = ts_{priv}^{(i)} H_1(t_{instance}) = s_i R H_1(t_{instance})$ to all system users using its private key.

Correspondingly, the *Dec* algorithm is modified as follows:

$M \leftarrow Dec(STs, Xs, T, U_T, params_{\text{tsmo}})$. The data receiver runs the *Dec* algorithm to recover the plaintext M. The following steps are required:

① At the decryption time T specified by the data sender, the data receiver randomly selects a set of valid time trapdoors STs from the time trapdoors published by N time servers, ensuring that $|STs| \geq t$. Xs is the corresponding set of identification numbers for the time servers. Calculate the main time trapdoor $S'_T = \sum_{S_T^{(i)} \in STs} S_T^{(i)} \prod_{x_j \in Xs, x_j \neq x_i} \frac{-x_j}{(x_i - x_j)}$.
② Calculates $K' = e(X, S'_T + U_T)$.
③ Recovers the plaintext $M = Y \oplus H_2(K')$.

Assume that the ciphertext is $C = \langle X, Y \rangle$, the decryption time is T, the set of valid time trapdoors is $STs = \{S_T^{(1)}, S_T^{(2)}, \ldots, S_T^{(t)}\}$, the corresponding set of identification numbers for the time servers is $Xs = \{x_1, x_2, \ldots, x_t\}$, and the user's trapdoor is U_T. The correctness of decryption is verified as follows:

$$\begin{aligned}
S'_T &= \sum_{S_T^{(i)} \in STs} S_T^{(i)} \prod_{x_j \in Xs, x_j \neq x_i} \frac{-x_j}{x_i - x_j} \\
&= s_1 R H_1(T) \cdot \frac{(0-x_2)(0-x_3)\cdots(0-x_t)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_t)} + \\
&\quad s_2 R H_1(T) \cdot \frac{(0-x_1)(0-x_3)\cdots(0-x_t)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_t)} + \cdots \\
&\quad + s_t R H_1(T) \cdot \frac{(0-x_1)(0-x_2)\cdots(0-x_{t-1})}{(x_t-x_1)(x_t-x_2)\cdots(x_t-x_{t-1})} \\
&= (s_1 \cdot \frac{(0-x_2)(0-x_3)\cdots(0-x_t)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_t)} + \\
&\quad s_2 \cdot \frac{(0-x_1)(0-x_3)\cdots(0-x_t)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_t)} + \cdots \\
&\quad + s_t \cdot \frac{(0-x_1)(0-x_2)\cdots(0-x_{t-1})}{(x_t-x_1)(x_t-x_2)\cdots(x_t-x_{t-1})}) \cdot R H_1(T) \\
&= s R H_1(T) \\
K' &= e(X, S'_T + U_T) \\
&= e(rP, s R H_1(T) + u H_1(T)) \\
&= e(r H_1(T), usk + tk) \\
&= K \\
Y \oplus H_2(K') &= Y \oplus H_2(K) \\
&= M \oplus H_2(K) \oplus H_2(K) \\
&= M
\end{aligned} \tag{10}$$

Security and efficiency analysis

Security analysis

The security properties of our proposed schemes are analyzed as follows.

(1) Data confidentiality. The attacker aims to illegally analyze key-related information necessary for decrypting the ciphertext before the specified time T. Assume that the attacker may attempt to crack the time server management organization's private key *sk* (SS-MSTRE$_2$ scheme is the parameter *sR*) and the user's private key *usk* through the time servers' public keys $ts_{pub}^{(i)}$ and the user's public key *upk*, this is equivalent to solving the ECDLP, which is currently considered infeasible, making it difficult for an attacker to effectively crack. Assume that time server TS_i stores many plaintext-ciphertext pairs, that is, one-way irreversible hash function calculation values of decryption time $H_1(T^*)$-time trapdoor S_T^* pair, it is difficult for an attacker to attack the time server TS_i's private key $ts_{priv}^{(i)}$ through the known-plaintext attack. The attacker can only obtain the corresponding private key by attacking t or more time servers to recover the main time trapdoor. However, this type of attack requires extremely high computational resources and time costs, making the probability of a polynomial-time attacker successfully breaking the ciphertext negligible.

(2) Verifiability. The time server obtains s_i through the *TS_KeyGen* algorithm, calculates $\mathbb{r}^* = \mathbb{H}_3(\sigma, s_i)$, and $U^* = \mathbb{r}^* \mathbb{P}$, and then compares U^* with the ciphertext C_i to detect whether s_i is legal and has not been tampered with. The time trapdoor is generated by combining a public hash function and the time server's private key with the security of the time server's private key depending on the ECDLP. When the time server TS_i sends the time trapdoor $S_T^{(i)}$ to the data receiver, the data receiver can also choose to use bilinear pairing technology to calculate and compare whether $e(ts_{pub}^{(i)}, H_1(T))$ and $e(P, S_T^{(i)})$ are equal, to detect whether $S_T^{(i)}$ is legal and has not been tampered with. The verifiability of the intermediate ciphertext data can effectively detect whether the original data is damaged due to noise and other factors when transmitting ciphertext data over a public network and can also resist attackers intercepting and tampering with data to a certain extent.

(3) Anti-advance decryption. Assume that a dishonest receiver wants to decrypt the data before the specified decryption time. As long as the ECDLP and BDH are still difficult problems at the current stage, it is a very difficult task, or almost impossible for the receiver

to decrypt the ciphertext based on the existing time server public keys, the specified decryption time T, its private key, and the public system parameters.

(4) Robust decryption with multiple time trapdoors. Our schemes employ Shamir secret sharing for key distribution to construct the main time trapdoor. The data receiver only needs to obtain time trapdoors that equal or exceed the threshold value to complete decryption. Consequently, even if some of the time servers fail or are attacked, the data receiver can still use a sufficient number of time trapdoors for decryption, significantly enhancing the reliability of the multiple time server TRE scheme.

We further provide proof that the SS-MSTRE$_1$ scheme is semantically secure against adaptive CPA [22].

Theorem 1 *Assume that adversary $\mathcal{A}$ has an advantage of ϵ in breaking the SS-MSTRE$_1$ scheme. Meanwhile, let the probability of challenger $\mathcal{B}$ overcoming the BDH assumption defined in Definition 2 be at least $\epsilon' = \epsilon/eq_Tq_{H_2}$, where e is the base of the natural logarithm, q_{H_2} is the maximum number of queries that $\mathcal{A}$ can make to the random oracle H_2, and q_T is the maximum number of queries that $\mathcal{A}$ can make to the time trapdoors of users and time servers.*

Proof
Assume that there is an adversary $\mathcal{A}$ with advantage ϵ in breaking the SS-MSTRE$_1$ scheme. $\mathcal{A}$ is limited to making no more than q_{H_2} queries to the random oracle H_2 and no more than q_T queries to the time trapdoors of user and time servers, where q_T and q_{H_2} are both positive. Let $\mathcal{B}$ be a challenger who can overcome the BDH assumption with a probability of at least $\epsilon' = \epsilon/eq_Tq_{H_2}$. Therefore, if the BDH assumption holds in G_1, then ϵ' can be considered negligible, and the advantage of $\mathcal{A}$ in breaking the SS-MSTRE$_1$ scheme can also be considered negligible. $\mathcal{B}$ simulates as the challenger and interacts with $\mathcal{A}$ as follows:

Preparation : Let G_1 be an ECDLP additive group of prime order q, G_2 be a DLP multiplicative group of prime order q, and let the bilinear mapping $e : G_1 \times G_1 \rightarrow G_2$ satisfy Definition 1. Give the challenger $\mathcal{B}$ the public parameters $P,\ P_1 = aP = uP + (\sum_{i=1}^{t} s_iP \cdot (\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p)),\ P_2 = bP$ and $P_3 = cP$. The goal of the challenger $\mathcal{B}$ is to output $v = e(P,P)^{abc} \in G_2$, where P is the generator of G_1 and $a, b, c \in \mathbb{Z}_p^*$.

Setup : Challenger $\mathcal{B}$ gives adversary $\mathcal{A}$ the data receiver's public key $upk = u$ and the time server's public key $ts_{pub}^{(i)} = s_iP$ (i=1,2,...,N).

Initialization : The adversary $\mathcal{A}$ outputs the target of the attack, a pair of decryption time points (T_0^*,T_1^*).

Phase 1 : The adversary $\mathcal{A}$ initiates $1, 2, \ldots, m$ queries, and the challenger $\mathcal{B}$ responds to each of them. The response process of the $\mathbb{i}$-th query is as follows:

The adversary $\mathcal{A}$ initiates a query to the random oracle H_1. Challenger $\mathcal{B}$ maintains an initially empty tuple list $H_1^{\text{list}} :< T_{\mathbb{j}}, h_{\mathbb{j}}, m_{\mathbb{j}}, n_{\mathbb{j}} >$. When the adversary $\mathcal{A}$ initiates a query for the time trapdoor to the random oracle H_1 at a time point $T_{\mathbb{i}}$, the challenger $\mathcal{B}$ responds as follows:

① If the tuple information $< T_{\mathbb{i}}, h_{\mathbb{i}}, m_{\mathbb{i}}, n_{\mathbb{i}} >$ containing $T_{\mathbb{i}}$ is already present in H_1^{list}, then the challenger $\mathcal{B}$ responds with $H_1(T_{\mathbb{i}}) = h_{\mathbb{i}} \in G_1$.
② If the tuple information $< T_{\mathbb{i}}, h_{\mathbb{i}}, m_{\mathbb{i}}, n_{\mathbb{i}} >$ containing $T_{\mathbb{i}}$ is not present in H_1^{list}, then the challenger $\mathcal{B}$ randomly generates a bit $n_{\mathbb{i}} \in \{0,1\}$ such that $\Pr[n_{\mathbb{i}} = 0] = 1/(q_T + 1)$.
③ The challenger $\mathcal{B}$ chooses a random number $m_{\mathbb{i}} \in \mathbb{Z}_q^*$. If $n_{\mathbb{i}} = 0$, then $\mathcal{B}$ calculates $h_{\mathbb{i}} = P_2 + m_{\mathbb{i}}P$, otherwise, calculates $h_{\mathbb{i}} = m_{\mathbb{i}}P$.
④ The challenger $\mathcal{B}$ adds the tuple $< T_{\mathbb{i}}, h_{\mathbb{i}}, m_{\mathbb{i}}, n_{\mathbb{i}} >$ to H_1^{list} and responds with $H_1(T_{\mathbb{i}}) = h_{\mathbb{i}}$. The value of $h_{\mathbb{i}}$ is uniformly distributed in G_1 and independent of the adversary $\mathcal{A}$.

Similarly, adversary $\mathcal{A}$ initiates a query to the random oracle H_2. the challenger $\mathcal{B}$ maintains an initially empty tuple list H_2^{list}, and responds as follows:

① When the adversary $\mathcal{A}$ queries H_2 for $H_2(K_{\mathbb{i}})$ and there is no information containing $K_{\mathbb{i}}$ in the list, the challenger $\mathcal{B}$ responds by choosing a new random value $V_{\mathbb{i}} \in \{0,1\}^{\log_2 q}$ and adding $(K_{\mathbb{i}}, V_{\mathbb{i}})$ to the tuple list H_2^{list}.
② If the tuple list H_2^{list} contains $(K_{\mathbb{i}}, V_{\mathbb{i}})$, then the challenger $\mathcal{B}$ takes $(K_{\mathbb{i}}, V_{\mathbb{i}})$ from H_2^{list} as the response value to the adversary $\mathcal{A}$.

When the adversary $\mathcal{A}$ initiates a time trapdoor query at a time point $T_{\mathbb{i}} \notin \{T_0^*, T_1^*\}$, the challenger $\mathcal{B}$ responds as follows:

① The challenger $\mathcal{B}$ runs the above H_1 query algorithm and obtains $H_1(T_{\mathbb{i}}) = h_{\mathbb{i}}$. Then, $\mathcal{B}$ makes the tuple $< T_{\mathbb{i}}, h_{\mathbb{i}}, m_{\mathbb{i}}, n_{\mathbb{i}} >$ as the corresponding element in the tuple list H_1^{list}.
② If $n_{\mathbb{i}} = 0$, the challenger $\mathcal{B}$ reports an error and terminates the entire simulation game.
③ If $n_{\mathbb{i}} = 1$, then $h_{\mathbb{i}} = m_{\mathbb{i}}P$.

Let $T_{u_{\mathbb{i}}} = m_{\mathbb{i}} \cdot upk$ and $T_{T_{\mathbb{i}}} = m_{\mathbb{i}} \cdot (\sum_{i=1}^{t} s_i P \cdot (\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p))$. We can also obtain $T_{u_{\mathbb{i}}} = uH_1(T_{\mathbb{i}})$ and $T_T = (\sum_{i=1}^{t} s_i \prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p) \cdot m_{\mathbb{i}} P = sH_1(T_{\mathbb{i}})$ by transforming the formulas. Here, $T_{u_{\mathbb{i}}}$ represents the legitimate user's time trapdoor at time point $T_{\mathbb{i}}$ and $T_{T_{\mathbb{i}}}$ represents the main time trapdoor at time point $T_{\mathbb{i}}$. The challenger $\mathcal{B}$ returns $T_{u_{\mathbb{i}}}$ and $T_{T_{\mathbb{i}}}$ to the adversary $\mathcal{A}$.

Challenge : The target of challenge for the adversary $\mathcal{A}$ is a pair of decryption time points (T_0^*, T_1^*). The challenger $\mathcal{B}$ produces the challenge ciphertext, and the response process is as follows:

① The challenger $\mathcal{B}$ performs two H_1 query algorithms to obtain h_0^* and $h_1^* \in G_1$, and obtains $H_1(T_0^*) = h_0^*$ and $H_1(T_1^*) = h_1^*$, which correspond to $< T_0^*, h_0^*, m_0^*, n_0^* >$ and $< T_1^*, h_1^*, m_1^*, n_1^* >$ respectively in H_1^{list}. If $n_0^* = n_1^* = 1$, the challenger $\mathcal{B}$ reports an error and terminates the entire simulation game.
② If either n_0^* or n_1^* is 0, the challenger $\mathcal{B}$ chooses a random number $\flat \in \{0,1\}$ such that $n_\flat^* = 0$.
③ The challenger $\mathcal{B}$ responds with the ciphertext $C_\flat^* = [P_3, J]$, where $J \in \{0,1\}^{log_2 q}$. Let

$$J = H_2\left(e\left(H_1\left(T_\flat^*\right), c \cdot upk\right) \cdot e\left(H_1\left(T_\flat^*\right), c \cdot \left(\sum_{i=1}^{t} s_i P \cdot \left(\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right) \tag{11}$$

Namely,

$$\begin{aligned} J &= H_2\left(e\left(H_1\left(T_\flat^*\right), upk\right)^c \cdot e\left(H_1\left(T_\flat^*\right), \left(\sum_{i=1}^{t} s_i P \cdot \left(\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)^c\right) \\ &= H_2\left(e\left(H_1\left(T_\flat^*\right), (u+s)P\right)^c\right) \\ &= H_2\left(e\left(P_2 + m_\flat^*, (u+s)P\right)^c\right) \\ &= H_2\left(e(P,P)^{\left(c(u+s)\left(b+m_\flat^*\right)\right)}\right) \end{aligned} \tag{12}$$

Therefore, $C_\flat^* = [P_3, J]$ is the true and valid ciphertext corresponding to the time $T_\flat^*$.

Phase 2 : The adversary $\mathcal{A}$ initiates time trapdoor queries for the user's time trapdoor and the time servers' time trapdoors from $m+1$ to num again. The challenger $\mathcal{B}$ responds in the same way as in *Phase* 1.

Guess : The adversary $\mathcal{A}$ outputs a guess of $\flat$, denoted as $\flat' \in \{0,1\}$, and guesses whether the ciphertext $C_\flat^*$ constructed by the challenger $\mathcal{B}$ in the *Challenge* phase is $Enc\left(upk, \left(\sum_{i=1}^{t} s_i P \cdot \left(\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right), T_0^*\right)$ or $Enc\left(upk, \left(\sum_{i=1}^{t} s_i P \cdot \left(\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right), T_1^*\right)$. At this point, the challenger $\mathcal{B}$ randomly chooses $(K_{\mathbb{j}}, V_{\mathbb{j}})$ from H_2^{list} and outputs $K/e\left(upk, \left(\sum_{i=1}^{t} s_i P \cdot \left(\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right), P_3\right)^{m_\flat^*}$ as a guess for $v = e(P,P)^{abc}$. If the adversary $\mathcal{A}$ has previously queried one of the items in $H_2\left(e\left(cH_1\left(T_0^*\right), upk + \left(\sum_{i=1}^{t} s_i P \left(\prod_{\substack{1 \le j \le t \\ j \ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right)$ or

$$H_2\left(e\left(cH_1\left(T_1^*\right), upk + \left(\sum_{i=1}^{t} s_iP\left(\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right),$$

then the tuple list H_2^{list} has a 1/2 probability of containing the tuple $(K_{\mathbb{j}}, V_{\mathbb{j}})$, where

$$\begin{aligned} K_{\mathbb{j}} &= H_2(e(cH_1(T_\flat^*), upk + (\sum_{i=1}^{t} s_iP \cdot (\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p)))) \\ &= H_2(e(P,P)^{c(u+s)(b+m_\flat^*)}) \end{aligned} \tag{13}$$

If the challenger $\mathcal{B}$ chooses the tuple $(K_{\mathbb{j}}, V_{\mathbb{j}})$ from H_2^{list}, then $K/e\left(upk + \left(\sum_{i=1}^{t} s_iP \cdot \left(\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right), P_3\right)^{m_\flat^*} = e(P,P)^{abc}$.

The entire simulation game ends at this point. Next, we calculate the probability value ϵ' of the challenger $\mathcal{B}$ correctly outputting $v = e(P,P)^{abc}$, and assume that the simulation game can continue to the *Guess* phase without any termination in between. To this end, we will begin by defining the following events:

Event E_1: the challenger $\mathcal{B}$ does not terminate the game during the phase when the adversary $\mathcal{A}$ queries the time trapdoor.

Event E_2: the challenger $\mathcal{B}$ does not terminate the game during the *Challenge* phase.

The probability of events E_1 and E_2 occurring is sufficiently high, and the following four claims are provided:

Claim 1 *During the phase when the adversary $\mathcal{A}$ queries the time trapdoor, the probability of the challenger $\mathcal{B}$ not terminating the game is at least $1/e$, with $\Pr[E_1] \ge 1/e$.*

Proof
Assume that the adversary $\mathcal{A}$ will not query the same time point twice. The response obtained from querying H_1 indicates that the probability of the challenger $\mathcal{B}$ terminating the game after one time trapdoor query by the adversary $\mathcal{A}$ is $1/(q_T+1)$. However, the adversary $\mathcal{A}$ can query the time trapdoor up to q_T times. Therefore, the probability of the challenger $\mathcal{B}$ not terminating the game after q_T time trapdoor queries by the adversary $\mathcal{A}$ is $(1-1/(q_T+1))^{q_T} \ge 1/e$.

Claim 2 *In the Challenge phase, the probability of the challenger $\mathcal{B}$ not terminating the game is at least $1/q_T$, with $\Pr[E_2] \ge 1/q_T$.*

Proof
Assume that the adversary $\mathcal{A}$ can generate a pair of designated decryption time points (T_0^*, T_1^*) with the property $n_0^* = n_1^* = 1$, then the challenger $\mathcal{B}$ terminates the game in the Challenge stage. Since the adversary $\mathcal{A}$ has not queried the time trapdoors for T_0^* and T_1^*, the values of n_0^* and n_1^* are not correlated with the adversary $\mathcal{A}$. Therefore, $\Pr[n_\flat^* = 0] = 1/(q_T+1)$ and $\Pr[n_0^* = n_1^* = 1] = (1-(1/(q_T+1)))^2 \le 1 - 1/q_T$. It follows that the probability of the challenger $\mathcal{B}$ not terminating the game in the Challenge phase is at least $1-(1-1/q_T) = 1/q_T$.

During the game process, the adversary $\mathcal{A}$ is not allowed to query the time trapdoors for T_0^* and T_1^*. Therefore, the events E_1 and E_2 are independent of each other, and it can be obtained that $\Pr[E_1 \cap E_2] \ge 1/eq_T$.

Assume that in the real attack game, the adversary $\mathcal{A}$ possesses the public key $upk = uP$ of the data receiver and the public keys $ts_{pub}^{(i)} = s_iP$ of the time servers $(i = 1, 2, ..., N)$. The adversary $\mathcal{A}$ sends a pair of decryption time points (T_0^*, T_1^*) to the challenger $\mathcal{B}$, and the challenger $\mathcal{B}$ generates a challenge ciphertext $C_\flat^* = [P_3, J]$ in response.

Claim 3 *In the real attack game, the adversary $\mathcal{A}$ has a probability of at least ϵ to initiate an H_2 query for either*
$$H_2\left(e\left(cH_1\left(T_0^*\right), upk + \left(\sum_{i=1}^{t} s_iP\left(\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right)$$
or $$H_2\left(e\left(cH_1\left(T_1^*\right), upk + \left(\sum_{i=1}^{t} s_iP\left(\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right).$$

Prior to presenting the proof, we will first provide the definitions for the following events:

Event E_3: In the real attack game, the adversary $\mathcal{A}$ does not initiate an H_2 query for either
$$H_2\left(e\left(cH_1\left(T_0^*\right), upk + \left(\sum_{i=1}^{t} s_iP\left(\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right)$$
or $$H_2\left(e\left(cH_1\left(T_1^*\right), upk + \left(\sum_{i=1}^{t} s_iP\left(\prod_{\substack{1\le j\le t\\ j\ne i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)\right).$$

Event E_4: In the *Guess* phase, the adversary $\mathcal{A}$ outputs the correct guess value $\flat'$, where $\flat = \flat'$.

Proof
When event E_3 occurs, it can be seen from the bit $\flat \in \{0,1\}$ that the ciphertext $C^*_\flat$ constructed by the challenger $\mathcal{B}$ is unrelated to the adversary $\mathcal{A}$. Therefore, the probability of event E_4 occurring is at most $\Pr[E_4]$ =1/2. In a real attack, the adversary $\mathcal{A}$ has an advantage ε, where $\Pr[E_4]$− (1/2)≥ ε. This also indirectly indicates that $\Pr[\neg E_3] \geq 2\varepsilon$. The proof process of $\Pr[\neg E_3] \geq 2\varepsilon$ is as follows:

$$\begin{aligned}\Pr[E_4] &= \Pr[E_4|E_3]\cdot\Pr[E_3] + \Pr[E_4|\neg E_3]\cdot\Pr[\neg E_3]\\ &\leq \Pr[E_4|E_3]\cdot\Pr[E_3] + \Pr[\neg E_3]\\ &\leq \frac{1}{2}\cdot\Pr[E_3] + \Pr[\neg E_3]\\ &\leq \frac{1}{2} + \frac{1}{2}\Pr[\neg E_3]\\ \Pr[E_4] &\geq \Pr[E_4|E_3]\cdot\Pr[E_3]\\ &\geq \frac{1}{2}\Pr[E_3]\\ &\geq \frac{1}{2} - \frac{1}{2}\Pr[\neg E_3]\end{aligned} \tag{14}$$

From formula (14), it is known that $\varepsilon \leq |\Pr[E_4]$−(1/2)| ≤(1/2)$\Pr[\neg E_3]$, which implies $\Pr[\neg E_3] \geq 2\varepsilon$.

Assume that the challenger $\mathcal{B}$ does not terminate the game, it means that in the simulated real game process, the adversary $\mathcal{A}$ has queried either $\mathrm{Enc}\left(upk, \left(\sum_{i=1}^{t} s_iP \cdot \left(\prod_{\substack{1\leq j\leq t\\ j\neq i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right), T_0^*\right)$ or $\mathrm{Enc}\left(upk, \left(\sum_{i=1}^{t} s_iP \cdot \left(\prod_{\substack{1\leq j\leq t\\ j\neq i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right), T_1^*\right)$. Therefore, $\Pr[\neg E_3] \geq 2\varepsilon$.

Claim 4 *The probability of the challenger $\mathcal{B}$ successfully solving the BDH assumption in the Challenge stage is ε/q_{H_2}.*

Proof
Assuming that the events described in Claim 3 occur, the value of one of the two possible cases of $e\left(cH_1\left(T^*_\flat\right), upk + \left(\sum_{i=1}^{t} s_iP \cdot \left(\prod_{\substack{1\leq j\leq t\\ j\neq i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)$ will be stored in the tuple list H_2^{list}. Therefore, in the Challenge phase, the challenger $\mathcal{B}$ will have a probability of $1/q_{H_2}$ to select the correct $e\left(H_1\left(T^*_\flat\right), upk + \left(\sum_{i=1}^{t} s_iP \cdot \left(\prod_{\substack{1\leq j\leq t\\ j\neq i}} \frac{0-x_j}{x_i-x_j} \bmod p\right)\right)\right)$ from H_2^{list}. If the challenger $\mathcal{B}$ does not terminate the simulation game, the probability of successfully solving the BDH assumption is ε/q_{H_2}. According to Claims 1 and 2, during the simulation game process, the probability that the challenger $\mathcal{B}$ does not terminate the game is at least ε/eq_T. And by Claim 4, if the challenger $\mathcal{B}$ does not terminate the simulation game, the probability of successfully solving the BDH assumption is ε/q_{H_2}.

In summary, through the above security simulation game, the probability of the challenger $\mathcal{B}$ successfully solving the BDH assumption is $\varepsilon/eq_Tq_{H_2}$. Theorem 1 is proven.

The above proof is equally applicable to the SS-MSTRE$_2$ scheme.

Efficiency analysis

In this section, we calculate the time consumption of our proposed SS-MSTRE schemes and only count the time consumption of *Enc*, *TS_Rel* and *Dec* algorithms (the two proposed SS-MSTRE schemes are basically consistent in operation), not count costs of the *TSMO_Setup*, *PKG_Setup*, *TempKey_Extract*, *KeySharing*, *TS_KeyGen*, *User_KeyGen* and *US_Rel* algorithms (all of these can be done in advance and are not included in the statistical scope). Among them, the supersingular elliptic curve of the finite field F_p (p is a 512-bit prime number) is defined as $y^2 = x^3 + 1(\bmod p)$, the prime order q is 160 bits, and the bilinear mapping adopts Tate pairs. BP stands for bilinear pairing operation; PM$_{ec}$ and PA$_{ec}$ represent point multiplication and point addition operations in group G_1; Add, Sub, Mul and Div represent modular addition, modular subtraction, modular multiplication, and modular division operations in $\mathbb{Z}_q^*$; H_1 represents a hash function that maps a binary string composed of 0 and 1 of any length to an element in G_1; H_2 represents a hash function that maps an element G_2 to a binary string composed of 0 and 1 of log_2q length.

We implement the above basic operations based on the open source large number operation function library MIRACL in cryptography and uses the approximate ratio method relative to the PM$_{ec}$ basic operations to record the time consumed by other basic operations and obtain the relative time-consuming table of basic operations, as shown in Table 2. The running environment is Intel(R) Core(TM) i5-7500 CPU 3.40GHz processor, 64-bit PC host, 8GB memory, and Microsoft visual studio 2017. 987654321 is used as the random

Table 2 Calculation cost of related basic operations relative to the PM_{ec} operation

Basic operation	Notation	Relative cost
Bilinear pairing	BP	3.7870
Point multiplication in G_1	PM_{ec}	1
Point addition operation in G_1	PA_{ec}	0.0074
Modular addition in $\mathbb{Z}_q^*$	Add	0.0005
Modular subtraction in $\mathbb{Z}_q^*$	Sub	0.0005
Modular multiplication in $\mathbb{Z}_q^*$	Mul	0.0008
Modular division in $\mathbb{Z}_q^*$	Div	0.0029
H_1 function:$\{0,1\}^* \to G_1$	H_1	0.3214
H_2 function:$G_2 \to \{0,1\}^{log_2 q}$	H_2	0.0784

number seed, and after running the program, one PM_{ec} operation takes about 1.5208 ms.

In the *Enc* algorithm, the data sender needs to complete the following operations: two PM_{ec} and one H_1 for rP and $rH_1(T)$, one BP and one PA_{ec} for $K' = e(rH_1(T), upk + tk)$ (tk can be precalculated and is not counted here), one H_2 for $M \oplus H_2(K')$; in the *TS_Rel* algorithm, the time server needs to complete the following operations: one H_1 and one PM_{ec} for $S_T^{(i)} = ts_{priv}^{(i)} \cdot H_1(T)$; in the *Dec* algorithm, the data receiver needs to complete the following operations: $t \cdot (t-1) \cdot$ Sub, $2t \cdot (t-2) \cdot$ Mul, t Div, t PM_{ec}, and $(t-1)$ PA_{ec} for $\sum_{S_T^{(i)} \in STs} S_T^{(i)} \prod_{x_j \in Xs, x_j \neq x_i} \frac{-x_j}{(x_i - x_j)}$. Additionally, one BP and one PA_{ec} for $K' = e(X, S_T' + U_T)$, and one H_2 for $M = Y \oplus H_2(K')$.Therefore, we compare the calculation time of the MTSTRE scheme in literature [30] with our SS-MSTRE schemes, as shown in Table 3.

It can be seen from Table 3 that the differences in computation time consumption between the MTSTRE scheme in literature [30] and our two SS-MSTRE schemes are mainly reflected in the *Enc* and *Dec* algorithms. In the *Enc* algorithm, the MTSTRE scheme in literature [30] requires N PA_{ec}, while our SS-MSTRE schemes only require one PA_{ec}; in the *Dec* algorithm, the MTSTRE scheme in literature [30] also needs N PA_{ec}, while our SS-MSTRE schemes need to complete $t \cdot ((t-1) \cdot \text{Sub} + 2(t-2) \cdot \text{Mul} + \text{Div} + PM_{ec} + PA_{ec})$ operations when using lagrange interpolation polynomial to reconstruct the main time trapdoor. Assume that there are m data senders in a certain application scenario. To compare the calculation cost of the two schemes, the calculation time of the MTSTRE scheme in the literature [30] may be expressed as follows:

$$\begin{aligned}\Pi_1 &= mEnc_t + TS\_Rel_t + mDec_t \\ &= m(0.0074N + 6.1868) + 1.3214 + m(0.0074N + 3.8654)\end{aligned} \tag{15}$$

the calculation of the SS-MSTRE schemes is expressed as follows:

$$\begin{aligned}\Pi_2 &= mEnc_t + TS\_Rel_t + mDec_t \\ &= 6.1942m + 1.3214 + m(0.0021t^2 + 1.0066t + 3.8654)\end{aligned} \tag{16}$$

Generally speaking, the value range of the secret sharing threshold t belongs to the interval $[\lfloor \frac{N}{2} \rfloor + 1, N]$ and the different setting of the threshold t will also affect the computational efficiency of our schemes. We will add a detailed application example to illustrate the case in Application example section.

Application example

Suppose the tenderer A is conducting a bidding process for a specific project, with the bid opening scheduled at "8:00 AM on March 8, 2023". Five bidders $(B_1, B_2, B_3, B_4, B_5)$ are invited to participate in the bidding, and the number of time servers is set to 10. Tenderer A establishes a time trapdoor threshold value of t for decrypting the submitted bids. We need to evaluate the computational costs of the *Enc* algorithm for the five bidders, the *TS_Rel* algorithm for the ten time servers, and the *Dec* algorithm for tenderer A to decrypt the five bids. The resulting comparison of the computational overhead in the sealed bidding application scenario is shown in Table 4.

When the threshold value t is set to 6, 7, and 8, the computation time of the proposed schemes in this paper is 82.1954, 87.3649, and 92.5554, respectively. Although these computational costs are higher than those of the MTSTRE scheme proposed in reference [30], the difference is negligible from a practical application perspective. Specifically, on an ordinary computer (such as the device environment used in this paper), the proposed schemes consume only an additional 0.04543(s), 0.05329(s), and 0.06119(s) compared to the MTSTRE scheme, making the difference imperceptible at the human perception level.

Table 3 Comparison of time consumption between MTSTRE scheme and our SS-MSTRE schemes

Scheme	Enc	TS_Rel	Dec	Total
MTSTRE	$2PM_{ec} + BP + N \cdot PA_{ec} + H_1 + H_2$	$PM_{ec} + H_1$	$BP + N \cdot PA_{ec} + H_2$	$0.0148N+11.3736$
SS-MSTRE	$2PM_{ec} + BP + PA_{ec} + H_1 + H_2$	$PM_{ec} + H_1$	$t \cdot ((t-1) \cdot Sub + 2(t-2) \cdot Mul + Div + BP + H_2 + PM_{ec} + PA_{ec})$	$0.0021t^2 + 1.0066t + 11.3810$

Table 4 Comparison of time consumption in sealed bidding application scenario

Scheme	Enc	TS_Rel	Dec	Total
MTSTRE	31.304	1.3214	19.6970	52.3224
SS-MSTRE($t = 6$)	30.9710	1.3214	49.9030	82.1954
SS-MSTRE($t = 7$)	30.9710	1.3214	55.0725	87.3649
SS-MSTRE($t = 8$)	30.9710	1.3214	60.2630	92.5554

Conclusions

This paper proposes a multiple time servers SS-MSTRE model based on Shamir secret sharing and presents two secure construction schemes. Our proposed schemes enable the receiver to recover EHR using any number of time trapdoors greater than or equal to a predefined threshold value when the preset decryption time arrives, which addresses the issue of single-point failure commonly found in traditional multiple time servers TRE schemes. We conduct security analysis of our schemes, provide a semantic secure proof against adaptive CPA based on the random oracle model, and employ the MIRACL big number arithmetic library to experimentally validate the efficiency of our schemes.

In future work, we will consider introducing other entities to complete the research on the TRE scheme with time-limited update key calculation and release together with the time server. Unlike the research idea of simply increasing the number of time servers, we aim to design other algorithms to construct a new encryption mechanism that solves the single point of failure problem in single time server TRE schemes.

Acknowledgements
The authors express their appreciation to the National Key Research and Development Program, the National Natural Science Foundation of China, the Fundamental Research Funds for the Central Universities of China, the Natural Science Foundation of Tianjin, the Key Specialized Research and Development Program of Henan Province, the Basic Higher Educational Key Scientific Research Program of Henan Province, and the Innovation Training Program for College Students of Henan province.

Authors' contributions
Ke Yuan put forward the main idea of the schemes, conceived and designed the experiments, analyzed the data, authored or reviewed drafts of the article, and approved the final draft. Ziwei Cheng performed the experiments, analyzed the data, performed the computation work, authored or reviewed drafts of the article, and approved the final draft. Keyan Chen prepared figures and tables, authored or reviewed drafts of the article, and approved the final draft. Bozhen Wang and Junyang Sun performed the experiments, performed the computation work, prepared figures and tables. Sufang Zhou proposed suggestions, analyzed the data and approved the final draft. Chunfu Jia analyzed the data, authored or reviewed drafts of the article, and approved the final draft.

Funding
This work was supported by the National Key Research and Development Program under Grant 2018YFA0704703, the National Natural Science Foundation of China under Grant 61972073, 61972215 and 62172238, the Fundamental Research Funds for the Central Universities of China, the Natural Science Foundation of Tianjin under Grant 20JCZDJC00640, the Key Specialized Research and Development Program of Henan Province under Grant 222102210062 and 222102210007, the Henan Province Science Foundation of Young Scholars under Grant 242300420678, the Basic Higher Educational Key Scientific Research Program of Henan Province under Grant 22A413004, the Innovation Training Program for College Students of Henan province under Grant 202310475143.

Availability of data and materials
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
The research has consent by all authors and there is no conflict.

Competing interests
The authors declare no competing interests.

Received: 27 February 2024 Accepted: 24 May 2024
Published online: 12 June 2024

References
1. Shi S, He D, Li L, Kumar N, Khan MK, Choo KKR (2020) Applications of blockchain in ensuring the security and privacy of electronic health record systems: A survey. Comput Secur 97:101966. https://doi.org/10.1016/j.cose.2020.101966
2. Liu Y, Yu W, Ai Z, Xu G, Zhao L, Tian Z (2023) A blockchain-empowered federated learning in healthcare-based cyber physical systems. IEEE Trans Netw Sci Eng 10(5):2685–2696. https://doi.org/10.1109/TNSE.2022.3168025
3. Keshta I, Odeh A (2021) Security and privacy of electronic health records: Concerns and challenges. Egypt Inf J 22(2):177–183. https://doi.org/10.1016/j.eij.2020.07.003
4. Khoda Parast F, Sindhav C, Nikam S, Izadi Yekta H, Kent KB, Hakak S (2022) Cloud computing security: A survey of service-based models. Comput Secur 114:102580. https://doi.org/10.1016/j.cose.2021.102580
5. Sandhu A (2022) Big data with cloud computing: Discussions and challenges. Big Data Min Analytics 5:32–40. https://doi.org/10.26599/BDMA.2021.9020016
6. Liu Y, Zhang C, Yan Y, Zhou X, Tian Z, Zhang J (2023) A semi-centralized trust management model based on blockchain for data exchange in iot system. IEEE Trans Serv Comput 16(2):858–871. https://doi.org/10.1109/TSC.2022.3181668
7. May T (1992) Timed-release crypto. http://www.hks.net.cpunks/cpunks-0/1560.html. Accessed 2 Mar 2022
8. Cheon JH, Hopper N, Kim Y, Osipkov I (2008) Provably secure timed-release public key encryption. ACM Trans Inf Syst Secur 11(2). https://doi.org/10.1145/1330332.1330336
9. Baird L, Mukherjee P, Sinha R (2022) i-tire: Incremental timed-release encryption or how to use timed-release encryption on blockchains? pp 235–248. https://doi.org/10.1145/3548606.3560704
10. Rivest RL, Shamir A, Wagner DA (1996) Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology (MIT). http://people.csail.mit.edu/rivest/pubs/RSW96.pdf
11. Mahmoody M, Moran T, Vadhan SP (2011) Time-lock puzzles in the random oracle model. In: Advances in Cryptology-crypto-Cryptology Conference, vol 6841. pp 39–50. https://doi.org/10.1007/978-3-642-22792-9_3
12. Bitansky N, Goldwasser S, Jain A, Paneth O, Vaikuntanathan V, Waters B (2016) Time-lock puzzles from randomized encodings. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science. ITCS '16. Association for Computing Machinery, New York, pp 345–356. https://doi.org/10.1145/2840728.2840745

13. Liu J, Jager T, Kakvi SA, Warinschi B (2018) How to build time-lock encryption. Des Codes Crytography 86:2549–2586. https://doi.org/10.1007/s10623-018-0461-x
14. Lai WJ, Hsueh CW, Wu JL (2019) A fully decentralized time-lock encryption system on blockchain. In: 2019 IEEE International Conference on Blockchain (Blockchain). pp 302–307. https://doi.org/10.1109/Blockchain.2019.00047
15. Hiraga D, Hara K, Tezuka M, Yoshida Y, Tanaka K (2021) Security definitions on time-lock puzzles. In: Hong D (ed) Information Security and Cryptology – ICISC 2020, vol 12593. Springer International Publishing, Cham, pp 3–15. https://doi.org/10.1007/978-3-030-68890-5_1
16. Chvojka P, Jager T, Slamanig D, Striecks C (2021) Versatile and sustainable timed-release encryption and sequential time-lock puzzles (extended abstract), vol 12973. Springer, Cham, pp 64–85. https://doi.org/10.1007/978-3-030-88428-4_4
17. Chan ACF, Blake IF (2005) Scalable, server-passive, user-anonymous timed release cryptography. pp 504–513. https://doi.org/10.1109/ICDCS.2005.72
18. Hwang YH, Yum DH, Lee PJ (2005) Timed-release encryption with pre-open capability and its application to certified e-mail system. In: Proceedings of the 8th International Conference on Information Security, ISC'05. Springer-Verlag, Berlin, pp 344–358. https://doi.org/10.1007/11556992_25
19. Hristu-Varsakelis D, Chalkias K, Stephanides G (2008) A versatile secure protocol for anonymous timed-release encryption. J Inf Assur Secur 2:80–88
20. Choi G, Vaudenay S (2019) 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, vol 11897. Springer-Verlag, Berlin, pp 167–179. https://doi.org/10.1007/978-3-030-39303-8_13
21. Namasudra S (2019) An improved attribute-based encryption technique towards the data security in cloud computing. Concurr Comput Pract Experience 31:4364–4364. https://doi.org/10.1007/978-3-030-39303-8_13
22. Yuan K, Wang Y, Zeng Y, Ouyang W, Li Z, Jia C, Peng H (2021) Provably secure security-enhanced timed-release encryption in the random oracle model. Sec Commun Netw 2021. https://doi.org/10.1155/2021/5593363
23. Yuan K, Cao H, Zhang S, Zhai C, Du X, Jia C (2023) A tamper-resistant timed secure data transmission protocol based on smart contract. Sci Rep 13:11510–11520. https://doi.org/10.1038/s41598-023-38136-3
24. Yuan K, Wang Z, Chen K, Zhou B, Li Z, Jia C (2024) Timed-release encryption anonymous interaction protocol based on smart contract 13(1):3–14. https://doi.org/10.1186/s13677-023-00536-1
25. Liu J, Garcia F, Ryan M (2015) Time-release protocol from bitcoin and witness encryption for sat. Korean Circ J 40:530–535
26. Li C, Palanisamy B (2018) "Decentralized Release of Self-Emerging Data using Smart Contracts," 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS), Salvador, Brazil, pp. 213–220. https://doi.org/10.1109/SRDS.2018.00033
27. Unruh D (2015) Revocable quantum timed-release encryption. J ACM 62(6). https://doi.org/10.1145/2817206
28. Chae SW, Kim JI, Park Y (2020) Practical time-release blockchain. Electronics 9(4):672–688. https://doi.org/10.3390/electronics9040672
29. Malavolta G, Thyagarajan SAK (2019) Homomorphic time-lock puzzles and applications. In: Boldyreva A, Micciancio D (eds) Advances in Cryptology - CRYPTO 2019, vol 11692. Springer International Publishing, Cham, pp 620–649
30. Yuan K, Cheng Z, Yang L, Yan Y, Jia C, He Y (2022) Research on timed-release encryption system based on multiple time servers. J Electron Inf Technol 44(12):4319–4327. https://doi.org/10.11999/JEIT211066
31. Shamir A (1979) How to share a secret. Commun ACM 22:612–613
32. Beimel A (2011) Secret-sharing schemes: A survey. In: Chee YM, Guo Z, Ling S, Shao F, Tang Y, Wang H, Xing C (eds) Coding and Cryptology. Springer Berlin Heidelberg, Berlin, pp 11–46
33. Porwal S, Mittal S (2021) A novel threshold secret sharing scheme for cp-abe: A secret sharing approach for cp-abe. In: Proceedings of the 2021 Thirteenth International Conference on Contemporary Computing, IC3-2021. Association for Computing Machinery, New York, pp 92–98. https://doi.org/10.1145/3474124.3474137
34. Al-Shaarani F, Gutub AAA (2021) Increasing participants using counting-based secret sharing via involving matrices and practical steganography. Arab J Sci Eng 47:2455–2477. https://doi.org/10.1007/s13369-021-06165-7
35. Liu Y, Zhang Y, Su S, Zhang L, Du X, Guizani M, Tian Z (2024) Blocksc: A blockchain empowered spatial crowdsourcing service in metaverse while preserving user location privacy. IEEE J Sel Areas Commun 42(4):880–892. https://doi.org/10.1109/JSAC.2023.3345416
36. Joux A (2002) The weil and tate pairings as building blocks for public key cryptosystems. In: Proceedings of the 5th International Symposium on Algorithmic Number Theory, ANTS-V. Springer-Verlag, Berlin, pp 20–32

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.