Hindawi
Security and Communication Networks
Volume 2021, Article ID 5593363, 10 pages
https://doi.org/10.1155/2021/5593363

Research Article

Provably Secure Security-Enhanced Timed-Release Encryption in the Random Oracle Model

Ke Yuan[1,2], **Yahui Wang**[1,2], **Yingming Zeng**[3], **Wenlei Ouyang**[1], **Zheng Li**[1] and **Chunfu Jia**[4]

[1]*School of Computer and Information Engineering, Henan University, Kaifeng 475004, China*
[2]*International Joint Research Laboratory for Cooperative Vehicular Networks of Henan, Henan University, Kaifeng 475004, China*
[3]*Beijing Institute of Computer Technology and Applications, Beijing 100854, China*
[4]*College of Cybersecurity, Nankai University, Tianjin 300350, China*

Correspondence should be addressed to Zheng Li; lizheng@henu.edu.cn

Received 19 February 2021; Accepted 14 May 2021; Published 27 May 2021

Academic Editor: Hao Peng

Cryptographic primitive of timed-release encryption (TRE) enables the sender to encrypt a message which only allows the designated receiver to decrypt after a designated time. Combined with other encryption technologies, TRE technology is applied to a variety of scenarios, including regularly posting on the social network and online sealed bidding. Nowadays, in order to control the decryption time while maintaining anonymity of user identities, most TRE solutions adopt a noninteractive time server mode to periodically broadcast time trapdoors, but because these time trapdoors are generated with fixed time server's private key, many "ciphertexts" related to the time server's private key that can be cryptanalyzed are generated, which poses a big challenge to the confidentiality of the time server's private key. To work this out, we propose a concrete scheme and a generic scheme of security-enhanced TRE (SETRE) in the random oracle model. In our SETRE schemes, we use fixed and variable random numbers together as the time server's private key to generate the time trapdoors. We formalize the definition of SETRE and give a provably secure concrete construction of SETRE. According to our experiment, the concrete scheme we proposed reduces the computational cost by about 10.8% compared to the most efficient solution in the random oracle model but only increases the almost negligible storage space. Meanwhile, it realizes one-time pad for the time trapdoor. To a large extent, this increases the security of the time server's private key. Therefore, our work enhances the security and efficiency of the TRE.

1. Introduction

Cryptographic primitive of timed-release encryption (TRE) [1, 2] requires the sender to set a specified time for the designative receiver to decrypt the secret message. With TRE, the sender encrypts a message and then sends to the receiver; before the decrypt time that the sender has set arrives, no one can decrypt this ciphertext. With the efforts of many distinguished scholars, TRE has developed into a basic cryptographic primitive, which can be combined with many other cryptographic primitives and applied to different fields, such as regularly posting on the social network [3, 4], edge caching [5], and ciphertext retrieval [6, 7].

According to the latest research, the TRE constructions have been extended from the mathematical problems [8–28] to the physical problems [29, 30] and the blockchain approach [31–34]. At present, a large number of TRE constructions are based on the mathematical problems. In practical terms, the most commonly used model is the noninteractive time server model. In this model, for the time server, neither the sender nor the receiver of the message interacts with it. The time server periodically broadcasts the time trapdoor. The receiver chooses the time trapdoor corresponding to the decryption time of the ciphertext to complete the decryption at the designated time.

However, in the current noninteractive TRE schemes, many time trapdoors related to the time server's private key will be generated. This will cause the attacker to have a certain amount of pairs (time, time trapdoor). Although the problems related to bilinear pairing are difficult to solve, the attacker can still adopt chosen-plaintext attack (CPA) or chosen-ciphertext attack (CCA) to attack the system, which seriously challenges the security of the private key of the time server. Thus, in this paper, we are working on this problem and trying to construct a solution.

1.1. Related Work. TRE was first proposed by May [2] in 1993 and then discussed in detail by Rivest et al. [1] in 1996. Most of the previous schemes can be divided into time-lock puzzles [1, 15, 18, 33] and agents categories. Most agent-class schemes use the time server as the agent, which are divided into interactive model [1, 8, 23, 24] and noninteractive model [9–14, 16, 17, 19–22, 25, 26, 28]. The time server method was originally constructed based on the quadratic residue problem [8]. After that, most of the proposed solutions are based on the assumption of the difficult bilinear pairing class problems, such as bilinear Diffie–Hellman (BDH) assumption [9–11, 13, 19, 21, 22, 24–28], bilinear Diffie–Hellman inversion (BDHI) assumption [12], and bilinear Diffie–Hellman exponent (BDHE) assumption [17].

In the solutions of the noninteractive server model, the time server's private key is used to perform an encryption-like operation on the hash function value of a time point T to generate a corresponding time trapdoor. Therefore, this model produces many pairs (plaintext, ciphertext) related to the private key of the time server. In response to this problem, we need to construct a new solution.

1.2. Our Contributions. We reexamine the noninteractive time server model in which the time server's private key is repeatedly used, resulting in many pairs (plaintext, ciphertext) related to the private key of the time server. In order to solve this problem, we construct a security-enhanced timed-release encryption (SETRE) solution based on the BDH assumption.

As we all know, in the operations of encryption and decryption, we use the private key k to encrypt the plaintext M and get the ciphertext $C = E_k(M)$ and use the private key k to decrypt the ciphertext C and get the plaintext $M = D_k(C)$. Similarly, we let the private key s and the hash function value $H(T)$ of a time point T perform some operations together to generate the corresponding time trapdoor $S_T = E_s(H(T))$; correspondingly, we can get $H(T) = D_s(S_T)$. In the above statement, S_T is equivalent to the ciphertext C, and $H(T)$ is equivalent to the plaintext M. If the attacker has many pairs (plaintext, ciphertext), then the security of the time server's private key will be greatly threatened.

Our SETRE schemes include a concrete scheme and a generic scheme. In our SETRE, the time server will use a random number x as the time server's session private key every time before publishing the time trapdoor. This session private key is combined with the time server's fixed private key to generate the time trapdoor $S_T = E_{(s,x)}(H(T))$ of our SETRE. Therefore, in our SETRE schemes, the secret private key involved in every generated time trapdoor is different. So, we can claim that our schemes realize one-time pad for the time trapdoor. In this case, the attacker can only get a pair of (plaintext, ciphertext) about the time point and its time trapdoor at most. Even if the attacker successfully obtains the private key of the time server corresponding to a time trapdoor, he cannot get the private key of the time server corresponding to other time trapdoors so that the time trapdoor cannot be generated in advance, which ensures that the receiver cannot decrypt in advance.

1.3. Organization. We begin by explaining what is SETRE. In Section 2, we give some cryptographic background and our generic public key encryption scheme. In Section 3, we formally define our SETRE and its simulation security game model. In Section 4, we present the concrete construction of SETRE and give its provably secure proof and the efficiency analysis. In Section 5, we provide the formal definition and construction of the generic SETRE and give its security analysis and efficiency analysis. Finally, we give the conclusion and future work.

2. Preliminary

We give a brief review of the bilinear pairing property, BDH assumption, and our generic public key encryption scheme that needs to be known in this section.

2.1. Properties of Bilinear Pairings. We give a form of bilinear pairings and their properties as described below.

Definition 1. Let G_1 be an elliptic curve discrete logarithm problem (ECDLP) additive group over a finite field, G_2 be a discrete logarithm problem (DLP) multiplicative group over a finite field, and the order of G_1, G_2 be a prime number q. The mapping e: $G_1 \times G_2 \longrightarrow G_2$ is a bilinear pairing mapping if e satisfies

(1) Bilinear property: given any $P, Q, R \in G_1$, the following operations hold:

$$e(P+Q,R) = e(P,R)e(Q,R),$$
$$e(P,Q+R) = e(P,Q)e(P,R). \quad (1)$$

(2) Nondegeneracy: suppose that the generator of group G_1 is P, then the generator of group G_2 is $e(P,P)$.

(3) Computability: given any two elements $P, Q \in G_1$, there must be an effective algorithm for calculating $e(P,Q)$.

2.2. BDH Assumption. Many cryptographic schemes are based on various difficult assumptions related to bilinear pairs, such as the (D)BDH assumption, (D)BDHI assumption, and (D)BDHE assumption [35, 36]. We now give the definition of the BDH assumption used in our SETRE schemes as follows.

Definition 2. Let G_1 be an ECDLP additive group over a finite field, P be the generator of G_1, G_2 be a DLP multiplicative group over a finite field, and the order of G_1, G_2 be a prime number q. Given $P, aP, bP, cP \in G_1^*$ (a, b, and c are evenly distributed in $\mathbb{Z}_q^*$), calculate $e(P,P)^{abc} \in G_2^*$. If $\Pr[\mathscr{A}(P, aP, bP, cP) = e(P,P)^{abc}] \geq \mathscr{E}$, then the advantage of the adversary $\mathscr{A}$ to solve the BDH assumption is $\mathscr{E}$, and $\mathscr{E}$ is negligible.

2.3. General Public Key Encryption Scheme. We simplify and abstract public key encryption (which has a certain characteristic) and only keep three phases which are initialization, encryption, and decryption; then, the general public key encryption (GPKE) scheme can be obtained.

Definition 3. $\mathscr{E}_{\text{GPKE}} = (\text{Setup}, \text{Enc}, \text{Dec})$ is the public key encryption algorithm, where

Setup: generates system public parameters and the user's public key and private key pairs $(\text{upk}, \text{usk}) = (uP, u)$ in which $u \in \mathbb{Z}_q^*$, $P \in G_1$ is a generator of G_1, and G_1 is an additive group

Enc: uses the user's public key uP to encrypt the plaintext to get the ciphertext $C_{\text{GPKE}} = \text{Enc}(M, uP)$

Dec: uses the user's private key u to decrypt the ciphertext to get the plaintext $M = \text{Dec}(C_{\text{GPKE}}, u)$

3. SETRE: Definitions

Suppose Bob is a social network user, and he wants to upload documents scheduled to be published regularly to the social network platform in advance so that he can pay attention to other things without worrying about this matter. And Bob does not want the social network platform to know in advance what he wants to publish. In this application scenario, Bob can use our SETRE solution to solve this problem securely and efficiently. Bob sends the following ciphertext of the document in advance with the designated decryption time:

$$C = \text{Enc}\left(M, ts_{\text{pub}}, ts_{\text{spub}}, \text{upk}, r, T\right), \tag{2}$$

where M is one of the documents planned to be released at a designated time point in the future, ts_{pub} and ts_{spub} are the time server's fixed public key and session public key, respectively, upk is the receiver's public key, r is a random number as a factor of freshness, and T is the designated decryption time. The social network platform can obtain the ciphertext of the document in advance but can only decrypt it in the future after the predetermined decryption time has arrived. We call such a cryptographic scheme noninteractive SETRE.

Definition 4. Our noninteractive concrete SETRE scheme includes three entities which are time server, sender, and receiver and polynomial-time randomized algorithm 7 tuples $\mathscr{E}_{\text{SETRE}} = (\text{Setup}, \text{TS\_KeyGen}, \text{User\_KeyGen}, \text{Enc}, \text{ST\_Rel}, \text{ST\_Rel}, \text{Dec})$, where

Setup: generates a public parameter params from a security parameter

TS_KeyGen: calculates and generates the fixed public/private key pair $(ts_{\text{pub}}, ts_{\text{priv}})$ and the session public/private key pair $(ts_{\text{spub}}, ts_{\text{spriv}})$ of the time server

User_KeyGen: calculates and generates the public/private key pair (upk, usk) of the system user

Enc: calculates the ciphertext C of the plaintext M, by using the public keys $ts_{\text{pub}}, ts_{\text{spub}}$, and upk, and a designated decryption time point T

ST_Rel: calculates a time server's time trapdoor S_T, by using the time server's fixed private key ts_{priv}, a designated decryption time point T, and its corresponding session private key ts_{spriv}

UT_Rel: calculates a user's time trapdoor U_T, by using the receiver's private key usk and a designated decryption time point T

Dec: calculates a plaintext M, by using a ciphertext C, the time server's time trapdoor S_T, and the receiver's time trapdoor U_T; or outputs a "reject" message

We use the simulation security game between the adversary $\mathscr{A}$ and the challenger $\mathscr{B}$ to formally define the security against the active adversary $\mathscr{A}$. The specific formal definition is as follows:

Preparation: public parameters are generated by the system.

Initialization: a pair of designated decryption time points T_0^* and T_1^* to be challenged is selected by the adversary $\mathscr{A}$.

Setup: the public parameters params and public keys upk, ts_{pub}, and ts_{spub} are generated by the challenger $\mathscr{B}$ and sent to the adversary $\mathscr{A}$.

Phase 1: the adversary $\mathscr{A}$ performs m queries of $q_1, q_2, \ldots, q_m$, where query q_i is one of the following:

(1) At any time point, the adversary $\mathscr{A}$ can perform queries of the random oracles H_1 and H_2. In response to H_1 and H_2 queries, the challenger $\mathscr{B}$ keeps two lists of H_1-list and H_2-list.

(2) Time trapdoor queries: time trapdoor query S_{T_i} and U_{T_i} of T_i where $T_i \notin \{T_0^*, T_1^*\}$. The challenger $\mathscr{B}$ responds by running algorithm ST_Rel and UT_Rel to generate the time trapdoors S_{T_i} and U_{T_i} corresponding to the designated decryption time point T_i. The challenger $\mathscr{B}$ then sends S_{T_i} and U_{T_i} to the adversary $\mathscr{A}$.

(3) Decryption queries: decryption query (C_i, T_i) for the designated decryption time point T_i. To decrypt the ciphertext (C_i, T_i), $\mathscr{B}$ runs algorithm Dec and uses the time trapdoors S_{T_i} and U_{T_i}. The challenger $\mathscr{B}$ then sends the decrypted plaintext M to the adversary $\mathscr{A}$.

These queries can be adaptive, which means that the response of q_i can be determined based on the responses of $q_1, q_2, \ldots, q_{i-1}$ previously queried.

Challenge: a pair of designated decryption time points T_0^* and T_1^* to be challenged is selected by the adversary $\mathscr{A}$. The challenger $\mathscr{B}$ selects a random bit $\flat \in \{0,1\}$, sets the ciphertext to be $(C_\flat^*, T_\flat^*)$, and then sends the challenge ciphertext $(C_\flat^*, T_\flat^*)$ to $\mathscr{A}$.

Phase 2: the adversary $\mathscr{A}$ performs other queries of $q_{m+1}, \ldots, q_{\text{num}}$, and the challenger $\mathscr{B}$ responds as shown in Phase 1.

Guess: in the end, the adversary $\mathscr{A}$ outputs a guess of $\flat' \in \{0,1\}$. If $\flat = \flat'$, then $\mathscr{A}$ wins the simulation security game.

We call such an adversary $\mathscr{A}$ an IND-sT-CCA adversary, and we can formally define the advantages of $\mathscr{A}$ attack our concrete SETRE scheme $\mathscr{E}$ as

$$\text{Adv}_{\mathscr{E},\mathscr{A}}^{\text{CCA}} = \left| P_r\left[\flat = \flat'\right] - \frac{1}{2} \right|. \tag{3}$$

Definition 5. Our concrete SETRE scheme $\mathscr{E}$ is said to be $(t, q_{H_2}, q_T, q_C, \varepsilon)$-selective designated decryption time, adaptive chosen-ciphertext secure if for any t-time IND-sT-CCA adversary $\mathscr{A}$ that performs at most $q_{H_2} H_2$ queries, q_T chosen designated decryption trapdoor queries, and $q_{\mathscr{C}}$ chosen decryption queries, we have that $\text{Adv}_{\mathscr{E},\mathscr{A}}^{\text{CCA}} < \varepsilon$. In other words, we call that $\mathscr{E}$ is $(t, q_{H_2}, q_T, q_C, \varepsilon)$ IND-sT-CCA secure.

We define our concrete SETRE scheme $\mathscr{E}$ to be IND-sT-CPA secure by simply disallowing the adversary $\mathscr{A}$ to perform decryption queries in the simulation security game described above.

Definition 6. Our concrete SETRE scheme $\mathscr{E}$ is said to be (t, q_T, ε)-selective designated decryption time, adaptive chosen-plaintext secure if $\mathscr{E}$ is $(t, q_{H_2}, q_T, 0, \varepsilon)$-selective designated decryption time, chosen-ciphertext secure. In other words, we call that $\mathscr{E}$ is $(t, q_{H_2}, q_T, \varepsilon)$ IND-sT-CPA secure.

4. Concrete Scheme of SETRE

We will attempt to propose a concrete scheme of SETRE based on the BDH assumption in the random oracle model.

4.1. Construction. The server-passive, scalable, user-anonymous TRE scheme proposed by Black and Chan (abbreviated as BC-TRE) laid the foundation of TRE. We now describe the concrete SETRE construction scheme. The scheme includes the following algorithm 7 tuples:

Setup: generates a public parameter $\text{params} = \{G_1, G_2, q, e, P, H_1, H_2, n\}$ from a security parameter k, where G_1 is an ECDLP additive group over a finite field, G_2 is a DLP multiplicative group over a finite field, and the order of G_1, G_2 is a prime number q, e: $G_1 \times G_1 \longrightarrow G_2$ is a bilinear mapping that satisfies Definition 1, $P \in G_1^*$ is the generator of additive group G_1, and H_1: $\{0,1\}^* \longrightarrow G_1$ and H_2: $G_2 \longrightarrow \{0,1\}^n$ (n is the length of the plaintext) are hash functions.

TS-KeyGen: the time server selects a random number $s \in \mathbb{Z}_q^*$ as the private key $ts_{\text{priv}} = s \in \mathbb{Z}_q^*$ of the time server and then calculates and generates the time server's public key $ts_{\text{pub}} = sP$. Similarly, the time server selects a random number set as the session private key set $\text{TS}_{\text{spriv}} = \{x_1, x_2, \ldots, x_l\} \in \mathbb{Z}_q^*$ of the time server and then calculates and generates the corresponding time server's session public key set $\text{TS}_{\text{spub}} = \{x_1 P, x_2 P, \ldots, x_l P\} \in G_1^*$ in which $l \approx 175200$ if we assume that a time trapdoor needs to be generated every half an hour and meet the demand for 10 consecutive years.

User-KeyGen: a user selects a random number $u \in \mathbb{Z}_q^*$ as its private key $\text{usk} = u \in \mathbb{Z}_q^*$ and then calculates and generates the system user's public key $\text{upk} = uP$.

Enc: the sender uses the public key $\text{upk}_r = uP$ of the receiver, the public key $ts_{\text{pub}} = sP$ of the time server, a designated decryption time point $T \in \{0,1\}^*$, and the time server's session public key $ts_{\text{spub}} = xP$ corresponding to the designated decryption time point $T \in \{0,1\}^*$ to encrypt the plaintext M as the following operations:

(1) Selects a random number $r \in \mathbb{Z}_q^*$ and calculates $U = rP$
(2) Calculates $S_{\text{pub}} = \text{upk} + ts_{\text{pub}} + ts_{\text{spub}} = uP + sP + xP = (u + s + x)P$
(3) Calculates $K = e(S_{\text{pub}}, rH_1(T)) = e(P, H_1(T))^{r(u+s+x)}$
(4) Outputs the ciphertext C

$$C = \langle U, V \rangle = \langle rP, M \oplus H_2(K) \rangle. \tag{4}$$

TS-Rel: the time server takes its own fixed private key $ts_{\text{priv}} = s$ and the session private key $ts_{\text{spriv}} = x$ of the current release time T and produces the time server's time trapdoor $S_T = (s + x)H_1(T)$.

UT_Rel: the receiver takes the private key $\text{usk} = u$ of his own and the current designated decryption time T and produces the user's time trapdoor $U_T = uH_1(T)$.

Dec: the receiver uses the time trapdoors S_T and U_T of the designated decryption time point $T \in \{0,1\}^*$ to decrypt the ciphertext $C = \langle U, V \rangle$ as the following operations:

(1) Calculates $K' = e(U, S_T + U_T)$
(2) Calculates $V \oplus H_2(K')$ to recover the corresponding plaintext M

Suppose C is the valid ciphertext; then, we have $U = rP$ and $V = M \oplus H_2(K)$. We can verify the correctness of the decryption as described in the following:

$$\begin{aligned} K' &= e\left(U_T + S_T, U\right) \\ &= e\left(uH_1(T) + (s+x)H_1(T), rP\right) \\ &= e\left((u+s+x)H_1(T), rP\right) \\ &= e\left(H_1(T), P\right)^{r(u+s+x)} \\ &= K, \\ V \oplus H_2\left(K'\right) &= V \oplus H_2(K) \\ &= M \oplus H_2(K) \oplus H_2(K) \\ &= M. \end{aligned} \tag{5}$$

4.2. Security of the Scheme. We give the proof that our SETRE scheme is noninteractive and semantically secure against CPA in the random oracle model, supposing that the BDH assumption is true [37].

Theorem 1. *Suppose that there is an adversary $\mathscr{A}$ who can break our SETRE scheme with the advantage of ϵ; then, a challenger $\mathscr{B}$, who can overcome the BDH problem with probability at least $\varepsilon' = \varepsilon / e q_T q_{H_2}$, is constructed, where e is the natural logarithm's base and q_T and q_{H_2} are the maximum number of times we assume the adversary $\mathscr{A}$ can query the time trapdoor and H_2 hash operation.*

Proof. Let $\mathscr{A}$ denote an adversary who has advantage ϵ to break the SETRE. Assume that $\mathscr{A}$ performs no more than q_{H_2} hash operation queries to H_2, no more than q_T user trapdoors, and the time server trapdoor queries, where q_T and q_{H_2} are positive. Let $\mathscr{B}$ denote a challenger who overcomes the BDH problem with probability no less than $\varepsilon' = \varepsilon / e q_{H_2} q_T$. Therefore, if the BDH assumption holds in G_1, then we can ignore ε'; furthermore, the advantage of $\mathscr{A}$ to break the SETRE can be ignored. And $\mathscr{B}$, who simulates as the challenger, will interact with adversary $\mathscr{A}$ as follows:

Preparation: let G_1 be an ECDLP additive group over a finite field, G_2 be a DLP multiplicative group over a finite field, the order of G_1, G_2 be a prime number q, e: $G_1 \times G_1 \longrightarrow G_2$ be a bilinear mapping that satisfies Definition 1, and $P \in G_1^*$ be the generator of additive group G_1. Give the challenger $\mathscr{B}$ the public parameter P, $P_1 = aP = uP + sP + xP$, $P_2 = bP$, and $P_3 = cP \in G_1$; the goal of $\mathscr{B}$ is to calculate the value of $v = e(P,P)^{abc} \in G_2$, where $a, b, c \in Z_q^*$.

Initialization: the adversary $\mathscr{A}$ outputs a pair of designated decryption time points T_0^* and T_1^* to be challenged.

Setup: the challenger $\mathscr{B}$ gives $\mathscr{A}$ the public keys $\mathrm{upk}_r = uP$, $ts_{\mathrm{pub}} = sP$, and $ts_{\mathrm{spub}} = xP$.

Phase 1: the adversary $\mathscr{A}$ initiates $1, \ldots, m$ queries, and $\mathscr{B}$ gives the response, respectively, where for the i-th query, $\mathscr{B}$'s response is described as follows:

(1) H_1 and H_2 queries: every point in time, the adversary $\mathscr{A}$ can perform queries of the random oracles H_1 and H_2. In response to H_1 queries, the challenger $\mathscr{B}$ keeps a list of quadruples $\langle T_j, h_j, m_j, n_j \rangle$, which we will call it the H_1-list and is initially set to be empty. If $\mathscr{A}$ performs a query of H_1 at a time point $T_i \in \{0,1\}^*$, then $\mathscr{B}$ gives the response as follows:

① If the query about T_i has been made before, then $\mathscr{B}$ takes $H_1(T_i) = h_i \in G_1$ as its response.

② If not, $\mathscr{B}$ chooses a new random bit $n_i \in \{0,1\}$ to satisfy $\Pr[n_i = 0] = 1/(q_s + 1)$.

③ $\mathscr{B}$ takes a random number $m_i \in Z_p$.

If $n_i = 0$ holds, $\mathscr{B}$ calculates $h_i \leftarrow P_2 + m_i \cdot P \in G_1$.

If $n_i = 1$ holds, $\mathscr{B}$ calculates $h_i \leftarrow m_i \cdot P \in G_1$.

④ $\mathscr{B}$ adds the quadruple $\langle T_i, h_i, m_i, n_i \rangle$ to the H_1-list and takes $H_1(T_i) = h_i \in G_1$ as its response to $\mathscr{A}$.

In the same way, $\mathscr{A}$ can perform a query to H_2 at any point in time. The H_2-list is initially set to be empty. $\mathscr{B}$ gives the response to the query on $H_2(K_i)$ by selecting a new random $V_i \in \{0,1\}^{\log_2 p}$ as the value of $H_2(K_i)$ for every new K_i and adding the tuple (K_i, V_i) to H_2-list. If H_2-list already contains (K_i, V_i), then $\mathscr{B}$ takes (K_i, V_i) from H_2-list and returns it to $\mathscr{A}$ as the response value.

(2) Time trapdoor queries: if the adversary $\mathscr{A}$ performs queries of the time trapdoor at a time point $T_i \notin \{T_0^*, T_1^*\}$, then the challenger $\mathscr{B}$ gives the response as follows:

① $\mathscr{B}$ runs the above H_1 query algorithm and obtains $H_1(T_i) = h_i \in G_1$ and makes $\langle T_i, h_i, m_i, n_i \rangle$ as the corresponding entry in H_1-list.

② If $n_i = 0$, then $\mathscr{B}$ aborts the simulation security game and admits failure.

③ If $n_i = 1$, we obtain $h_i = m_i \cdot P \in G_1$. Let $T_{u_i} = m_i \cdot \mathrm{upk}_r$ and $T_{T_i} = m_i \cdot (ts_{\mathrm{pub}} + ts_{\mathrm{spub}})$; then, we can transform them to get $T_{u_i} = uH_1(T_i)$ and $T_{T_i} = (s + x_i)H_1(T_i)$. Therefore, T_{u_i} is the correct and legal user time trapdoor of T_i, and T_{T_i} is the correct and legal time server trapdoor of T_i. $\mathscr{B}$ gives T_{u_i} and T_{T_i} to $\mathscr{A}$.

Challenge: the adversary $\mathscr{A}$ selects a pair of designated decryption time points (T_0^*, T_1^*) to be challenged. The challenger $\mathscr{B}$ produces the challenge ciphertext as follows:

① The challenger $\mathscr{B}$ runs the above H_1 query algorithm twice to obtain h_0^* and $h_1^* \in G_1$ which satisfy $H_1(T_0^*) = h_0^*$ and $H_1(T_1) = h_1^*$.

② For $i = 0, 1$, we let $\langle T_0^*, h_0^*, m_0^*, n_0^* \rangle$ and $\langle T_1^*, h_1^*, m_1^*, n_1^* \rangle$ to be the corresponding tuples on the H_1-list. If $n_0' = n_1' = 1$, then the challenger $\mathscr{B}$ aborts the simulation security game and admits failure.

③ Obviously, at least one of n_0^* and n_1^* must be equal to zero. $\mathscr{B}$ randomly takes $\flat \in \{0,1\}$ such that $n_\flat = 0$.

④ $\mathscr{B}$ takes the challenge ciphertext $C_\flat^* = [P_3, J]$ for random $J \in \{0,1\}^{\log_2 p}$ as its response. Obviously, this challenge implicitly defines $H_2(e(H_1(T_\flat^*), c \cdot \mathrm{upk}_r) \cdot e(H_1(T_\flat^*), c \cdot ts_{\mathrm{pub}}) \cdot e(H_1(T_\flat^*), c \cdot ts_{\mathrm{spub}\flat})) = J$. That is to say,

$$\begin{aligned} J &= H_2\Big(e\Big(cH_1(T_\flat^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}\flat}\Big)\Big) \\ &= H_2\left(e\left(P_2 + m_\flat^* P, (u+s+x_\flat)P\right)^c\right) \\ &= H_2\Big(e(P,P)^{c(u+s+x_\flat)(b+m_\flat^*)}\Big). \end{aligned} \tag{6}$$

It can be seen that $C_\flat^*$ is the corresponding valid and real ciphertext for $T_\flat^*$.

Phase 2: the adversary $\mathscr{A}$ performs other queries of $q_{m+1}, \ldots, q_{\mathrm{num}}$, and the challenger $\mathscr{B}$ responds as shown in Phase 1.

Guess: in the end, the adversary $\mathscr{A}$ outputs a guess of $\flat' \in \{0,1\}$ to indicate whether the challenge ciphertext $C_\flat^*$ is a valid ciphertext for $\mathrm{Enc}(\mathrm{upk}_r, ts_{\mathrm{pub}}, ts_{\mathrm{spub}0}, T_0^*)$ or $\mathrm{Enc}(\mathrm{upk}_r, ts_{\mathrm{pub}}, ts_{\mathrm{spub}1}, T_1^*)$. Now, the challenger $\mathscr{B}$ randomly selects a tuple (K_j, V_j) from the H_2-list and outputs $K/e(\mathrm{upk}_r, ts_{\mathrm{pub}}, ts_{\mathrm{spub}\flat}, P_3)^{m_\flat^*}$ as a guess of $e(P,P)^{abc}$. If $\mathscr{A}$ has ever inquired about one of $H_2(e(cH_1(T_0^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}0}))$ or $H_2(e(cH_1(T_1^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}1}))$, the H_2-list has a probability of 1/2 that contains (K_j, V_j), $K_j = H_2(e(cH_1(T_\flat^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}\flat}) = H_2(e(P,P)^{c(u+s+x_\flat)(b+m_\flat^*)}))$. If $\mathscr{B}$ takes this tuple (K_j, V_j) from the H_2-list, then $K/e(\mathrm{upk}_r, ts_{\mathrm{pub}} + ts_{\mathrm{spub}\flat,P_3})^{m_\flat^*} = e(P,P)^{\mathrm{abc}}$.

The whole security simulation game is completed here. Next, we calculate the value of ϵ' which is the lowest probability of $\mathscr{B}$ correctly outputting $e(P,P)^{abc}$. It is easy to know that the premise that it can correctly output its guess value of $e(P,P)^{abc}$ is that the game can continue to the guessing stage without terminating the game in the middle. Now, we analyze the possibility that $\mathscr{B}$ does not terminate the game while the game is in progress. For this purpose, we first give the definition of the following events:

$\mathscr{E}_0$: in the stage when the adversary $\mathscr{A}$ performs queries of the time trapdoor, the challenger $\mathscr{B}$ does not terminate the simulation security game

$\mathscr{E}_1$: in the challenge stage, the challenger $\mathscr{B}$ does not terminate the simulation security game

We first state that, as in [38], events $\mathscr{E}_1$ and $\mathscr{E}_2$ occur with a high enough probability. Next, we give the following three claims.

Claim 1 : in the stage when the adversary $\mathscr{A}$ performs queries of the time trapdoor, the probability that the challenger $\mathscr{B}$ does not terminate the simulation security game is $1/e$ at least. Thus, $P_r[\mathscr{E}_0] \geq 1/e$. □

Proof. When the adversary $\mathscr{A}$ queries for the time trapdoor of time points, for the sake of generality, we suppose that $\mathscr{A}$ does not query the same time trapdoor twice. A trapdoor (the user's time trapdoor or the time server's time trapdoor) query causes $\mathscr{B}$ to terminate the simulation security game with a probability of $1/(q_T+1)$; therefore, a trapdoor query does not cause $\mathscr{B}$ to terminate the game with a probability of $(1 - 1/(q_T+1))$. In addition, since the maximum number of times $\mathscr{A}$ can query the time trapdoor is q_T, the probability that the simulation security game will not be terminated after q_T queries is $(1-1/(q_T+1))^{q_T} \geq 1/e$ at least.

Claim 2: in the challenge stage, the probability that the challenger $\mathscr{B}$ does not terminate the simulation security game is $1/q_T$ at least. Thus, $P_r[\mathscr{E}_1] \geq 1/q_T$. □

Proof. If the adversary $\mathscr{A}$ can generate T_0^*, T_1^* with the property $n_0^* = n_1^* = 1$, then the challenger $\mathscr{B}$ will terminate the simulation security game during the challenge stage. Since $\mathscr{A}$ has not queried for the trapdoor for T_0^*, T_1^*, we have that n_0^*, n_1^* are independent of $\mathscr{A}$. Therefore, $P_r[n_\flat^* = 0] = 1/(q_T+1)$ for $\flat = 0, 1$, and then we have that $P_r[n_0^* = n_1^* = 1] = (1-1/(q_T+1))^2 \leq 1 - 1/q_T$. Therefore, there is a probability of at least $1/q_T$ that $\mathscr{B}$ does not terminate the game.

Since the adversary $\mathscr{A}$ is not allowed to query the time trapdoor of the designated decryption time T_0, T_1 during the game, the events $\mathscr{E}_0$ and $\mathscr{E}_1$ are independent of each other, so we can get $P_r[\mathscr{E}_0 \cap \mathscr{E}_1] \geq 1/eq_T$.

Assume that the adversary $\mathscr{A}$ has acquired the public keys $\mathrm{upk}_r = uP$, $ts_{\mathrm{pub}} = sP$, and $ts_{\mathrm{spub}} = xP$ in the actual attack game. The adversary $\mathscr{A}$ selects a pair of designated decryption time points (T_0^*, T_1^*) to be challenged. The challenger $\mathscr{B}$ produces the challenge ciphertext $C_\flat^* = [P_3, J]$ as a response. Therefore, we have the following Claim 3.

Claim 3: in the actual attack game, the adversary $\mathscr{A}$ has at least the probability of ϵ to perform an H_2 query for one of $H_2(e(cH_1(T_0^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}0}))$, $H_2(e(cH_1(T_1^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}1}))$.

Before giving the proof, we first give the definition of the following events:

$\mathscr{E}_2$: in the actual attack game, $\mathscr{A}$ does not query either $H_2(e(cH_1(T_0^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}0}))$ or $H_2(e(cH_1(T_1^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}1}))$

$\mathscr{E}_3$: in the guess stage, $\mathscr{A}$ outputs the guess $\flat'$ of $\flat$ satisfying $\flat = \flat'$ □

Proof. When $\mathscr{E}_2$ occurs, it is obvious that the bit $\flat \in \{0,1\}$ indicates whether $C_\flat^*$ is the challenge ciphertext corresponding to the designated decryption time, which has nothing to do with $\mathscr{A}$'s knowledge. Thus, the probability of $P_r[\mathscr{E}_3]$ is 1/2 at most. In the real attack game, because $\mathscr{A}$ has the advantage of ϵ, we have $|P_r[\mathscr{E}_3] - 1/2| \geq \varepsilon$ and

$P_r[\neg\mathscr{E}_2] \geq 2\varepsilon$. Now, we give the specific argument for the truth of $P_r[\neg\mathscr{E}_2] \geq 2\varepsilon$ as follows:

$$\begin{aligned} P_r[\mathscr{E}_3] &= P_r[\mathscr{E}_3|\mathscr{E}_2]P_r[\mathscr{E}_2] + P_r[\mathscr{E}_3|\neg\mathscr{E}_2]P_r[\neg\mathscr{E}_2] \\ &\leq P_r[\mathscr{E}_3|\mathscr{E}_2]P_r[\mathscr{E}_2] + P_r[\neg\mathscr{E}_2] \\ &\leq \frac{1}{2}P_r[\mathscr{E}_2] + P_r[\neg\mathscr{E}_2] \\ &\leq \frac{1}{2} + \frac{1}{2}P_r[\neg\mathscr{E}_2]P_r[\mathscr{E}_3] \\ &\geq P_r[\mathscr{E}_3|\mathscr{E}_2]P_r[\mathscr{E}_2] \\ &\geq \frac{1}{2}P_r[\mathscr{E}_2] \geq \frac{1}{2} - \frac{1}{2}P_r[\neg\mathscr{E}_2]. \end{aligned} \tag{7}$$

From the above two formulas, we know that $\epsilon \leq |P_r[\mathscr{E}_3] - (1/2)| \leq (1/2)P_r[\neg\mathscr{E}_2]$. Thus, we have $P_r[\neg\mathscr{E}_2] \geq 2\varepsilon$ in the actual attack game.

If the challenger $\mathscr{B}$ does not terminate the game, it means that, in the process of simulating the actual attack game, the adversary $\mathscr{A}$ has queried one of $H_2(e(cH_1(T_0^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub0}}))$, $H_2(e(cH_1(T_1^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub1}}))$. Thus, $P_r[\neg\mathscr{E}_2] \geq 2\epsilon$.

Claim 4: the probability that the challenger $\mathscr{B}$ can solve the BDH problem successfully in the guess stage is ϵ/q_{H_2}. □

Proof. Assuming the event of Claim 3 occurs, the value of one of the two cases of $e(cH_1(T_b^*), \mathrm{upk}_r + ts_{\mathrm{pub}} + ts_{\mathrm{spub}b})$ will be stored in the H_2-list. Consequently, in the guess stage, the challenger $\mathscr{B}$ has at least the probability of $1/q_{H_2}$ to select the correct pair from the H_2-list. Therefore, on the premise that $\mathscr{B}$ does not terminate the simulation game, the possibility that $\mathscr{B}$ can successfully solve the BDH problem is $\epsilon/(q_{H_2})$.

According to Claims 1 and 2, during the simulation game, the probability that the challenger $\mathscr{B}$ will not terminate the game is at least $1/eq_T$. And according to Claim 4, if $\mathscr{B}$ does not terminate the simulation security game, the probability that $\mathscr{B}$ can successfully solve the BDH problem is ϵ/q_{H_2}. Therefore, through the security simulation game of the aforementioned adversary $\mathscr{A}$ and challenger $\mathscr{B}$, the possibility of successfully solving the BDH problem is $\varepsilon/eq_Tq_{H_2}$. Thus, Theorem 1 is proved. □

4.3. Efficiency Analysis. We contrast between our SETRE scheme and two representative noninteractive time server TRE schemes: the classic BC-TRE scheme put forward by Blake and Chan [9] and the AnTRE scheme, which has highest efficiency up till now, put forward by Chalkias et al. [12].

We let BP be a notation of the bilinear pairing operation, PA_{ec} and PM_{ec} be a notation of point addition and point multiplication operations in G_1 separately. Let Exp_{ec} be a notation of the exponentiation operation in G_2 and Inv be a notation of the modular inverse operation in $\mathbb{Z}_q^*$. Let H_1 represent a hash function operation that maps binary strings of any length to an element in group G_1, H_2 represent the hash function operation that maps an element in group G_2 to a string of $\log_2^q$ length 0 and 1, and H_3 represent the hash function operation of mapping a binary string of any length to an element of $\mathbb{Z}_q^*$. Based on the MIRACL large integer library, we program and implement the basic operations described above, in which the relevant parameters are set as follows: the elliptic curve is a supersingular elliptic curve E: $y = x^3 + 1 \bmod p$ on the finite field F_p (p is a 512-bit large prime number), and its prime order q is a 160-bit prime number; the bilinear map uses the Tate pairing algorithm to map the aforementioned discrete logarithm subgroup on the elliptic curve to the discrete logarithm subgroup on F_{p^2}. The configuration of the running environment is as follows: Intel(R) Core(TM) i5-4210M @ 2.60 GHz microprocessors, 64 bit and 8 GB memory, Microsoft Visual Studio 2010. 987654321 is the seed that generates the associated random numbers. We take the calculation time of Exp_{ec} as the basic unit so that the calculation results are not related to the specific computer performance. We then calculate and record the ratio of the calculation time of each related basic operation in these schemes to the calculation time of Exp_{ec}, as shown in Table 1.

Table 1: Calculation cost of related basic operations relative to the Exp_{ec} operation.

Related basic operations	Notation	Relative cost
Bilinear pairing	BP	3.4457
Point addition in G_1	PA_{ec}	0.0072
Point multiplication in G_1	PM_{ec}	1
Exponentiation in G_2	Exp_{ec}	0.3220
Modular inverse in $\mathbb{Z}_q^*$	Inv	0.0030
Hash function: $\{0,1\}^* \longrightarrow G_1$	H_1	0.3368
Hash function: $G_1 \longrightarrow \{0,1\}^{\log_2^q}$	H_2	0.0782
Hash function: $\{0,1\}^* \longrightarrow \mathbb{Z}_q^*$	H_3	0.0030

In our SETRE scheme, the TS-Rel stage requires one PM_{ec} and one H_1 to calculate $S_T = (s + x)H_1(T)$, and the total calculation cost of the TS-Rel stage is 1.003. The Enc stage requires one PM_{ec} for rP, two PA_{ec} for S_{pub}, one H_1, one PM_{ec}, and one BP for $e(rH_1(T), S_{\mathrm{pub}})$, and one H_2 for $H_2(K)$, and the total calculation cost of the Enc stage is 5.875. The Dec stage requires one H_1, one PM_{ec}, one PA_{ec}, and one BP for $K' = e(U, S_T + uH_1(T))$ and one H_2 for $M \oplus H_2(K)$, and the total calculation cost of the Dec stage is 4.868. We sum up the calculation cost of the schemes of BC-TRE, AnTRE, and our SETRE as shown in Table 2. It should be pointed out that the hash functions H_1 and H_2 in the scheme of AnTRE are approximately equivalent to H_3 in Table 1, and the hash functions H_3 and H_4 in the scheme of AnTRE are approximately equivalent to H_2 in Table 1.

Table 2 shows that our SETRE scheme has improved by 32.4% and 10.8%, respectively, compared with the schemes of BC-TRE and AnTRE. In addition, in the aspect of security,

Table 2: Calculation cost of BC-TRE, AnTRE, and our SETRE.

Scheme	Phase		
	BC-TRE [9]	AnTRE [12]	Our SETRE
TS − Rel	$PM_{ec} + H_1 = 1.337$	$PM_{ec} + Inv + H_3 = 1.006$	$PM_{ec} + H_1 = 1.337$
Enc	$3BP + 2PM_{ec} + H_3 = 12.340$	$4PM_{ec} + PA_{ec} + Exp_{ec} + BP + 2H_2 + 2H_3 = 7.934$	$2PM_{ec} + 2PA_{ec} + H_1 + BP + H_2 = 5.875$
Dec	$BP + Exp_{ec} + H_1 + H_2 = 4.183$	$BP + PM_{ec} + 2H_2 + H_3 = 4.605$	$H_1 + PM_{ec} + PA_{ec} + BP + H_2 = 4.868$
Total	17.86	13.55	12.08

our SETRE scheme realizes one-time pad for the time trapdoor. Therefore, compared with the previous schemes, our SETRE greatly improves the security of the time server's private key. In terms of storage, we need 160 bits of storage space for each private key of the time server and 1024 bits for each public key of the time server. Therefore, if it is assumed that the time server broadcasts a time trapdoor every half an hour and needs to store the session time server private key and public key for 10 years, then the additional storage space required by our scheme is $24 * 2 * 365 * 10 * (1024 + 160)$ bit ≈ 24.7 MB, which only adds almost negligible storage burden to the time server.

5. Generic Scheme of SETRE

We will attempt to propose a generic scheme of SETRE based on GPKE and call it generic SETRE, abbreviated as GSETRE.

5.1. Formal Definition. We now formalize the definition of our GSETRE scheme.

Definition 7. Our GSETRE scheme includes three entities which are time server, sender, and receiver and a polynomial-time randomized algorithm 6 tuples $\mathscr{E}_{\mathrm{GSETRE}} = (\mathrm{Setup}, \mathrm{TS\_KeyGen}, \mathrm{User\_KeyGen}, \mathrm{Enc}, \mathrm{Rel}, \mathrm{Dec})$, where

Setup: generates a public parameter params from a security parameter

TS_KeyGen: calculates and generates the fixed public/private key pair $(ts_{\mathrm{pub}}, ts_{\mathrm{priv}})$ and the session public/private key pair $(ts_{\mathrm{spub}}, ts_{\mathrm{spriv}})$ of the time server

User_KeyGen: calculates and generates the public/private key pair $(\mathrm{upk}, \mathrm{usk})$ of the system user

Enc: inputs $M, \mathrm{upk}, T, ts_{\mathrm{pub}}$, and ts_{spub} that correspond to the designated decryption time point T to the Enc algorithm of $\mathscr{E}_{\mathrm{GPKE}}$ and outputs the ciphertext C_{GSETRE}

Rel: given the private key ts_{priv} of the time server, a designated decryption time point T, and its corresponding session private key ts_{spriv} and produces a time server's time trapdoor S_T

Dec: inputs S_T and usk into the Dec algorithm of $\mathscr{E}_{\mathrm{GPKE}}$ and outputs plaintext M or $\perp$

5.2. Construction. We construct a $\mathscr{E}_{\mathrm{GSETRE}}$ scheme by introducing $\mathscr{E}_{\mathrm{SETRE}}$ into $\mathscr{E}_{\mathrm{PKE}}$. $\mathscr{E}_{\mathrm{GSETRE}}$ includes the following algorithm 6 tuples:

Setup: this algorithm is consistent with the Setup algorithm of our concrete $\mathscr{E}_{\mathrm{SETRE}}$ scheme

TS_KeyGen: this algorithm is consistent with the TS_KeyGen algorithm of our concrete $\mathscr{E}_{\mathrm{SETRE}}$ scheme

User_KeyGen: this algorithm is consistent with the User_KeyGen algorithm of our concrete $\mathscr{E}_{\mathrm{SETRE}}$ scheme

Enc: the sender uses $\mathrm{upk}_r = uP$ of the receiver, $ts_{\mathrm{pub}} = sP$ of the time server, a designated decryption time point $T \in \{0,1\}^*$, and $ts_{\mathrm{spub}} = xP$ corresponding to the designated decryption time point $T \in \{0,1\}^*$ to encrypt the plaintext M as the following operations:

(1) Uses upk_r to encrypt the plaintext M and calculates $\mathscr{E}_{\mathrm{PKE}}$'s ciphertext $C_{\mathrm{GPKE}} = \mathrm{Enc}(M, uP)$.
(2) Chooses a function $f(\cdot)$ randomly and calculates $U = f(\cdot) \cdot P$.
(3) Uses C_{GPKE}, ts_{pub}, ts_{spub}, and T to calculate

$$\begin{aligned} V &= C_{\mathrm{GPKE}} \cdot e\left(H_1(T), f(\cdot) \cdot \left(ts_{\mathrm{pub}} + ts_{\mathrm{spub}}\right)\right) \\ &= C_{\mathrm{GPKE}} \cdot e\left(H_1(T), f(\cdot) \cdot (s + x)P\right). \end{aligned} \tag{8}$$

(4) Outputs $\mathscr{E}_{\mathrm{GSETRE}}$'s ciphertext $C_{\mathrm{GSETRE}} = \langle U, V \rangle = \langle f(\cdot) \cdot P, \mathrm{Enc}(M, uP) \cdot e(H_1(T), f(\cdot) \cdot (s+ x)P) \rangle$.

Rel: this algorithm is consistent with the ST_Rel algorithm of our concrete $\mathscr{E}_{\mathrm{SETRE}}$ scheme

Dec: the receiver uses the time trapdoors S_T of the designated decryption time point $T \in \{0,1\}^*$ and the private key $\mathrm{usk}_r = u$ of the receiver to decrypt the ciphertext $C_{\mathrm{GSETRE}} = \langle U, V \rangle$ as the following operations:

(1) Calculates $C'_{\mathrm{GPKE}} = (V / e(S_T, U))$
(2) Calculates $\mathrm{Dec}(C'_{\mathrm{GPKE}}, u)$ to recover the corresponding plaintext M

Suppose C_{GSETRE} is the valid ciphertext; then, we have $U = f(\cdot) \cdot P$ and $V = C_{\mathrm{GPKE}} \cdot e(H_1(T), f(\cdot) \cdot (s + x)P)$. We

can verify the correctness of the decryption as described in the following:

$$\begin{aligned} C'_{\mathrm{GPKE}} &= \frac{V}{e(S_T, U)} \\ &= \frac{V}{e((s+x)H_1(T), f(\cdot)\cdot P)} \\ &= \frac{C_{\mathrm{GPKE}}\cdot e(H_1(T), f(\cdot)\cdot (s+x)P)}{e(H_1(T), f(\cdot)\cdot (s+x)P)} \\ &= C_{\mathrm{GPKE}}\mathrm{Dec}(C'_{\mathrm{GPKE}}, u) \\ &= \mathrm{Dec}(C_{\mathrm{GPKE}}, u) = M. \end{aligned} \tag{9}$$

5.3. Security and Efficiency Analysis. From the perspective of security, since the $\mathscr{E}_{\mathrm{GSETRE}}$ scheme is obtained by introducing $\mathscr{E}_{\mathrm{SETRE}}$ into the $\mathscr{E}_{\mathrm{GPKE}}$ scheme, which is equivalent to encapsulating the $\mathscr{E}_{\mathrm{GPKE}}$ scheme's ciphertext, the security of the $\mathscr{E}_{\mathrm{GPKE}}$ scheme will be enhanced after introducing $\mathscr{E}_{\mathrm{SETRE}}$. Firstly, the decryption operation needs to decrypt the $\mathscr{E}_{\mathrm{SETRE}}$'s ciphertext to get the ciphertext of the $\mathscr{E}_{\mathrm{GPKE}}$ scheme. However, the decryption of $\mathscr{E}_{\mathrm{SETRE}}$ requires a valid time trapdoor, and the attacker cannot construct the required time trapdoor without knowing the time server's private key and session private key. Secondly, decrypting the $\mathscr{E}_{\mathrm{GPKE}}$ ciphertext requires the private key of the legitimate receiver.

From the perspective of efficiency, compared with the $\mathscr{E}_{\mathrm{GPKE}}$ scheme, the $\mathscr{E}_{\mathrm{GSETRE}}$ scheme adds other additional operations in the encryption and decryption process, which inevitably leads to a decrease in efficiency. However, when using the idea of the general scheme to construct a concrete scheme, the parameters of the $\mathscr{E}_{\mathrm{SETRE}}$ scheme can be integrated into the same logical step of the $\mathscr{E}_{\mathrm{GPKE}}$ scheme as far as possible, so as to minimize the decline of efficiency. In addition, in terms of storage space, the time server only needs to add a small amount of storage space, as described in the above section.

6. Summary and Outlook

With the purpose of enhancing TRE security, a concrete SETRE scheme and a generic SETRE scheme based on the BDH assumption in the random oracle model are put forward. In our SETRE schemes, the time server uses a different session key to generate an "encryption-like" trapdoor at different time points. This operation uses the idea of one-time pad for the generation of each time trapdoor, which prevents the time trapdoor from being known in advance due to the leakage of the time server's private key and thus prevents the ciphertext from being decrypted in advance.

To ensure the anonymity of each system user identity to the time server, most current TRE solutions use broadcast to distribute time trapdoors. If time trapdoors are broadcast in a coarse-grained manner, many users may not have corresponding time trapdoors for the specified decryption time. In order to meet the time trapdoor specified by the user as far as possible, it is required to broadcast the time trapdoors with fine granularity, but this would waste communication resources. Therefore, designing a TRE that can support the specified arbitrary release time, anonymize the user identity, and prevent the time server from denial-of-service attacks will be a very practical and challenging task in the future. In addition, we will explore the combination of TRE with other cryptographic primitives, such as order-revealing encryption [39], so that more scenarios can have the function of controlling the decryption time.

Data Availability

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

Disclosure

A preliminary version of this paper appears in the "International Symposium on Security and Privacy in Social Networks and Big Data-6th International Symposium SocialSec 2020, Tianjin, China," in September 26-27, 2020, based on https://link.springer.com/book/10.1007%2F978-981-15-9031-3?page=3#toc.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by grants from the National Key R&D Program of China (no. 2018YFA0704703) and the National Natural Science Foundation of China (nos. 61802111, 61972073, and 61972215).

References

[1] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," Technical Report MIT/LCS/TR-684, MIT LCS Tech, Cambridge, MA, USA, 1996.

[2] T. May, *Timed-release Crypto*, Unpublished manuscript, 1993.

[3] B. Alexander, D. Levshun, N. Krasilnikova et al., "Determination of young generation's sensitivity to the destructive stimuli based on the information in social networks," *Journal of Internet Services and Information Security (JISIS)*, vol. 9, no. 3, pp. 1–20, 2019.

[4] M. Kolomeets, A. Benachour, D. El Baz et al., "Reference architecture for social networks graph analysis," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 10, no. 4, pp. 109–125, 2019.

[5] X. Wei, J. Liu, Y. Wang, C. Tang, and Y. Hu, "Wireless edge caching based on content similarity in dynamic environments," *Journal of Systems Architecture*, vol. 115, Article ID 102000, 2021.

[6] W. Yu, S. Lv, X. Guo, Z. Liu, Y. Huang, and B. Li, "Fsse: forward secure searchable encryption with keyed-block chains," *Information Sciences*, vol. 500, pp. 113–126, 2019.

[7] X. Jiang, X. Ge, Y. Jia, F. Kong, X. Cheng, and H. Rong, "An efficient symmetric searchable encryption scheme for cloud

storage," *Journal of Internet Services and Information Security (JISIS)*, vol. 7, no. 2, pp. 1–18, 2017.

[8] M. M. Casassa, H. Keith, and S. Martin, "The hp time vault service: exploiting ibe for timed release of confidential information," in *Proceedings of the 12th International Conference on World Wide Web*, pp. 160–169, ACM, Budapest, Hungary, May 2003.

[9] A. C.-F. Chan and I. F. Blake, "Scalable, server-passive, user-anonymous timed release cryptography," in *Proceedings of 25th IEEE International Conference on Distributed Computing Systems. ICDCS 2005*, pp. 504–513, IEEE, Colombus, OH, USA, June 2005.

[10] Y. H. Hwang, D. H. Yum, and P. J. Lee, "Timed-release encryption with pre-open capability and its application to certified e-mail system," *Lecture Notes in Computer Science*, Springer, Berlin, Germany, pp. 344–358, 2005.

[11] W. D. Alexander and Q. Tang, "Revisiting the security model for timed-release encryption with pre-open capability," in *Proceedings of International Conference on Information Security*, pp. 158–174, Springer, Valparaiso, CL, USA, October 2007.

[12] K. Chalkias, D. Hristu-Varsakelis, and G. Stephanides, "Improved anonymous timed-release encryption," in *Proceedings of European Symposium on Research in Computer Security ESORICS 2007*, pp. 311–326, Springer, Dresden, Germany, September 2007.

[13] J. H. Cheon, N. Hopper, Y. Kim, and I. Osipkov, "Provably secure timed-release public key encryption," *ACM Transactions on Information and System Security*, vol. 11, no. 2, pp. 1–44, 2008.

[14] A. Fujioka, Y. Okamoto, and T. Saito, "Generic construction of strongly secure timed-release public-key encryption," in *Proceedings of Australasian Conference on Information Security and Privacy*, pp. 319–336, Springer, Melbourne, Australia, July 2011.

[15] M. Mohammad, T. Moran, and V. Salil, "Time-lock puzzles in the random oracle model," in *Proceedings of Advances in Cryptology-CRYPTO 2011, LNCS 6841*, pp. 39–50, Springer, Santa Barbara, CA, USA, August 2011.

[16] J. Xiong, F. Li, J. Ma, X. Liu, Z. Yao, and P. S. Chen, "A full lifecycle privacy protection scheme for sensitive data in cloud computing," *Peer-to-peer Networking and Applications*, vol. 8, no. 6, pp. 1025–1037, 2015.

[17] K. Yuan, Z. Liu, C. Jia, J. Yang, and S. Lv, "Public key timed-release searchable encryption in one-to-many scenarios," *Acta Electronica Sinica*, vol. 43, no. 4, pp. 760–768, 2015.

[18] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, and B. Waters, "Time-lock puzzles from randomized encodings," in *Proceeding of Acm Conference on Innovations in Theoretical Computer Science*, pp. 345–356, ACM, Cambridge, MA, USA, January 2016.

[19] S.-Y. Huang, C.-I. Fan, and Y.-F. Tseng, "Enabled/disabled predicate encryption in clouds," *Future Generation Computer Systems*, vol. 62, pp. 148–160, 2016.

[20] S. Namasudra, "An improved attribute-based encryption technique towards the data security in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 9, p. e4364, 2017.

[21] W. Chen, Y. Wang, Z. Qin, and X. Liu, "Research on timed access of sensitive data based on dual encryption," *Journal of University of Electronic Science and Technology of China*, vol. 46, no. 3, pp. 588–593, 2017.

[22] C.-I. Fan, J.-C. Chen, S.-Y. Huang, J.-J. Huang, and W.-T. Chen, "Provably secure timed-release proxy conditional reencryption," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2291–2302, 2017.

[23] S. Y. Patil and J. N. Archana, "Conjunctive keyword search with designated tester and timing enabled proxy reencryption in health cloud," *International Journal for Innovative Research in Science and Technology*, vol. 4, no. 3, pp. 78–85, 2017.

[24] Q. Huang, Y. Yang, and J. Fu, "Secure data group sharing and dissemination with attribute and time conditions in public cloud," *IEEE Transactions on Services Computing*, vol. 99, 2018.

[25] Y. Watanabe and J. Shikata, "Timed-release computational secret sharing and threshold encryption," *Designs, Codes and Cryptography*, vol. 86, no. 1, pp. 17–54, 2018.

[26] H. Cao, K. Yuan, Y. Wang, Y. Yan, L. Zhou, and X. Chai, "Bidding model based on timed-release encryption and blockchain," *Journal of Henan University (Natural Science)*, vol. 49, no. 2, pp. 210–217, 2019.

[27] G. Choi and S. Vaudenay, "Timed-release encryption with master time bound key (extended)," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 10, no. 4, pp. 88–108, 2019.

[28] J. Hong, K. Xue, Y. Xue et al., "TAFC: time and attribute factors combined access control for time-sensitive data in public cloud," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 158–171, 2020.

[29] D. Unruh, "Revocable quantum timed-release encryption," in *Proceedings of Advances in Cryptology-EUROCRYPT 2014*, pp. 129–146, Springer, Copenhagen, Denmark, May 2014.

[30] T. Wang, Y. He, and L. Li, "New timed-release encryption based on indistinguishability obfuscation," *Application Research of Computers*, vol. 34, no. 9, pp. 2795–2798, 2017.

[31] L. Jia, F. Garcia, and M. Ryan, "Time-release protocol from bitcoin and witness encryption for sat," *Korean Circulation Journal*, vol. 40, no. 10, pp. 530–535, 2015.

[32] C. Li and B. Palanisamy, "Decentralized release of self-emerging data using smart contracts," in *Proceedings of 2018 IEEE 37th Symposium on Reliable Distributed Systems*, pp. 213–220, IEEE, Salvador, Brazil, October 2018.

[33] L. Jia, T. Jager, S. A. Kakvi, and W. Bogdan, "How to build time-lock encryption," *Designs, Codes and Cryptography*, vol. 86, pp. 2549–2586, 2018.

[34] W. Lai Jr, H. Chih-Wen, and J.-L. Wu, "A fully decentralized time-lock encryption system on blockchain," in *Proceedings of 2019 IEEE International Conference on Blockchain*, pp. 302–307, IEEE, Atlanta, GA, USA, July 2019.

[35] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (kase) for group data sharing via cloud storage," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2374–2385, 2016.

[36] H. Tsuchida, T. Nishide, and E. Okamoto, "Expressive ciphertext-policy attribute-based encryption with fast decryption," *Journal of Internet Services and Information Security (JISIS)*, vol. 8, no. 4, pp. 37–56, 2018.

[37] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of Advances in Cryptology-EUROCRYPT 2004*, pp. 506–522, Springer, Interlaken, Switzerland, May 2004.

[38] J.-S. Coron, "On the exact security of full domain hash," in *Proceedings of Advances in Cryptology-CRYPTO 2000*, pp. 229–235, Springer, Santa Barbara, CF, USA, August 2000.

[39] Z. Liu, L. Jin, S. Lv et al., "Encodeore: reducing leakage and preserving practicality in order-revealing encryption," *IEEE Transactions on Dependable and Secure Computing*, vol. XX, 2020.