A time-aware searchable encryption scheme for EHRs

Yousheng Zhou a,b,*, Xiaofeng Zhao a, Siling Liu a, Xingwang Long a, Wenjun Luo a,b

a College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China
b School of Cyber Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China

A R T I C L E   I N F O

Keywords:
- Searchable encryption
- Timed-release
- Designated server
- Privacy preserving
- EHRs

A B S T R A C T

Despite the benefits of EHRs (Electronic Health Records), there is a growing concern over the risks of privacy exposure associated with the technologies of EHR storing and transmission. To deal with this problem, a time-aware searchable encryption with designated server is proposed in this paper. It is based on Boneh’s public key encryption with keyword search and Rivest’s timed-release cryptology. Our construction has three features: the user cannot issue a keyword search query successfully unless the search falls into the specific time range; only the authorized user can generate a valid trapdoor; only the designated server can execute the search. Applying our scheme in a multi-user environment, the number of the keyword ciphertexts would not increase linearly with the number of the authorized users. The security and performance analysis shows that our proposed scheme is secure and more efficient than the existing similar schemes.

1. Introduction

Electronic Health Records (EHRs) have been widely adopted to promote the operational efficiency of hospitals [1] nowadays. This practice also brings great benefits to patients. For instance, it enables them to ask for support and advice from their physicians remotely via home computers or mobile devices rather than going to the clinics or hospitals in person in every occasion. Moreover, the advances in information and communication technologies make intelligent healthcare possible. For example, making use of the mass health records stored in many hospitals as input to an analysis system, doctors can find out the inner principals of certain diseases, which are critical for their treatment. In recent years, as many hospitals are equipped with more advanced clinical instruments and computer aided devices, a huge amount of data will be generated and need to be stored and maintained, which would form a heavy burden to the IT support department of these hospitals. As a result, many hospitals are apt to outsource the data storage and maintenance tasks to a third party, such as cloud providers [2,3]. Therefore, there have been wide concerns over the issue of privacy exposure as the EHRs could be exposed to those third party servers and even to unauthorized parties [4]. Encryption is a common method to prevent the exposure of personal health records [5,6], and sensible records could be encrypted with certain encryption algorithm before outsourcing. However, the retrieval of encrypted records remains to be a challenge to those schemes, because the health records could be retrieved by other authorized systems and users. To achieve a tradeoff betweeen the confidentiality and the availability, the searchable public key encryption can be considered as a practical solution for this problem.

Public key Encryption with Keyword Search (PEKS), proposed by Boneh et al. [8], is a scheme that allows one to search a document containing a particular keyword by providing a trapdoor, while other parties cannot learn anything except for the trapdoor. In this scheme, a secure channel between the client and the server must be established to deliver the trapdoor in its security model, which is very impractical because building such a secure channel is quiet expensive. In 2008, Baek et al. [9] presented a new PEKS scheme with a designated server (dPEKS), in which the requirement of a secure channel was removed. To prevent the keyword-guessing attacks in the dPEKS schemes, Hyun et al. [12] investigated the security model for dPEKS and introduced the concept of trapdoor indistinguishability. Similar proposals appeared in Refs. [13-15].

dPEKS schemes are suitable for EHRs, because the personal health records are private and need to be protected, meanwhile they might be used by different systems and physicians, and even might be outsourced to third parties. Actually, not all patients’ records have the same requirements for privacy. Although some personal health records are confidential and strictly constrained in usage at present, they might be legally shared and retrieved by authorized servers and users in the future. A dPEKS scheme can perform search at a specified time in the future other than at present, thus is suitable for such a scenario. It’s a pity that...
Fig. 1. T-dPEKS scenario.

there is no similar dPEKS scheme with this feature by now. Since the
search on encrypted health records are restricted to specific servers at a
specific time, the combination of dPEKS and timed-release would be a
promising solution. The concept of timed-release was first proposed by
May in 1993 [16], and a detailed discussion was made by Rivest et al. in
1996 [17]. It is an encrypt mechanism that only after the specific time T
can the user decrypt the encrypted data. Some applications of the
timed-release encryption were discussed in Refs. [18–22].

In this paper, we concentrate on constructing a Time-aware dPEKS
scheme (T-dPEKS), which can perform search at a specified time in the
future by authorized servers. As far as we know, there is no similar dPEKS
scheme for EHRs up to now.

Our proposed scheme is multi-user setting oriented, in which each
health record owner firstly encrypts his or her sensitive health records
using certain encryption algorithm, then encrypts the corresponding
keywords for the records, and then uploads the encrypted content to the
storage center, such as a medical cloud. When a record owner encrypts
his or her record, he/she can assign which user can retrieve and which
server can search these encrypted records. Meanwhile, the owner will
also specify a specific time T so that only after this specific time can the
encrypted health records be properly retrieved and decrypted by a record
user. The detailed procedure can be found in Fig. 1, where health records
owners are, S1, S2 are servers to perform the search, and U1, U2, U3 are
health records users. Assume T1 is the system time in this scenario, only
U1 can successfully get the encrypted records R1, which contains the
keyword K1. And U1 can decrypt the encrypted records from S1 with K1,
which is pre-allocated. In this way, the privacy of health records can be
preserved effectively.

The rest of this paper is organized as follows: firstly, we provide some
mathematical preliminaries in Section 2; then we define T-dPEKS and its
security model in Section 3. Then, the concrete construction of the pro-
sposed scheme is illustrated in Section 4. Subsequently, the security
analysis and comparison are presented in Section 5. Finally, the paper is
concluded in Section 6.

2. Preliminaries

As our scheme is based on the bilinear parings, this section presents a
brief review on the related basic knowledge about bilinear parings.

2.1. Bilinear parings

Assume G1 and G2 are two multiplicative cyclic groups with the same
prime order p, then a bilinear pairing e : G1 × G1 → G2 is defined under
the following conditions:

1. Bilinearity: For all P, Q ∈ G1 and a, b ∈ Zp, e(aP, bQ) = e(P, Q)ab.
2. Non-degeneracy: There exist P, Q ∈ G1, such that e(P, Q) ≠ 1.
3. Computability: For all P, Q ∈ G1, there exists an efficient algorithm to
compute e(P, Q).

2.2. Bilinear Diffie-Hellman problem (BDH)

Let g be a generator of G1. The BDH problem is defined as follows:
Given g, g^a, g^b ∈ G1 as input, compute e(g, g)^ab ∈ G2. BDH is called
intractable if all polynomial time algorithms can solve the BDH problem
with a negligible advantage.

2.3. Hash Diffie-Hellman assumption (HDH)

Let g be a generator of G1, let Ib be a security number and H : 
{o, 1}^* → {0, 1}^l be a Hash Function. The HDH problem in G1 is defined
as follows: given g, g^a, g^b, H(g^c) as input, output “yes” if a · b = c, and “no”
otherwise. An algorithm A that outputs b’ ∈ {0, 1} has an advantage ε in
solving the HDH problem in G1 if

Pr[A(g, g^a, g^b, H(g^c))] = "yes": g ← G1,

a, b ← Zp

|Pr[A(g, g^a, g^b, H(g^c))] − Pr[A(g, g^a, g^b, \eta) = "yes"]|

≥ ε

We say that the HDH assumption holds in G1 if no t-time algorithm has
an advantage at least ε in solving the HDH problem in G1.

3. Security definition

This section gives the model of T-dPEKS and its security definitions.
3.1. Definition of T-dPEKS

A T-dPEKS is a tuple of algorithms as follows: (Setup(λ), serverKeyGen(param), userKeyGen(param), T – dPEKS(param, pk_u, w, pk_s, T). Trapdoor(param, sk_u, w, pk_s). Test(C, sk_s, Tw, T).

Setup(λ): Input a security parameter λ, and output the global parameter set param.

serverKeyGen(param): Input the system global parameters param, and output the public key and private key (pk_s, sk_s) for server S.

userKeyGen(param): Input the system global parameters param, and output the public key and private key (pk_u, sk_u) for user U.

T – dPEKS(param, pk_u, w, pk_s, T): Input the system global parameters param, the public keys of the authorized users pk_u(j = 1, ..., n), the keyword w, the specific time T, and the public key of the designated server pk_s; and output an encryption C of the keyword w.

Trapdoor(param, sk_u, w, pk_s): Input the global parameters param, the private key of user u, the public key of the designated server pk_s, and the keyword w; and output a search trapdoor Twu.

Test(C, sk_s, Tw, T): Input the ciphertext C, the private key of the designated server sk_s and a search trapdoor Tw; then it outputs ‘yes’ if w = w’, and ‘no’ otherwise, where C = T – dPEKS(param, pk_u, w’, pk_s, T).

3.2. Security of T-dPEKS ciphertext

We need to ensure that the trapdoor C = T – dPEKS(param, pk_u, w, pk_s, T) will leak nothing about w unless Tw is available. Next we will define the security against an adversary who is capable of getting Tw for any keyword w he/she wants. That is the adversary who cannot distinguish the ciphertext of the keyword w0 and keyword w1, without the trapdoors of both keywords. The security model is based on the security model proposed and defined by Boneh et al. [8] by using a game as follows:

Definition (1). Ciphertext Indistinguishability Game.

Given the security parameter λ and the public parameter param, a challenger B and an adversary A can interact by using the following game:

Setup: Challenger B runs the Setup(λ) algorithm to generate the global parameters param and the public keys of the authorized users pk_u, where (j = 1, ..., n), and then sends these values to adversary A.

Trapdoor query: Adversary A can adaptively ask the challenger for the trapdoor Tw of any keyword w ∈ {0, 1}λ for user j.

Challenge step: Adversary A chooses between w0, w1, which he/she wants to challenge firstly, and then challenger B picks w2 randomly, where b ∈ {0, 1}, and sends T – dPEKS(param, pk_u, w2, pk_s, T) to A.

The constraint is: Tw0 and Tw1 haven’t been queried before.

More trapdoor queries: Adversary A can adaptively ask the challenger for the trapdoor Tw for any keyword w ∈ {0, 1}λ for user j.

The restriction is: The keyword w should not be w0 or w1.

Adversary A outputs b as a guess for b. If b = b’, the adversary wins the game. The advantage of A in breaking the scheme is defined as follows:

\[
Adv_{T-dPEKS,A}^{\text{ind-cpa}}(\lambda) = \Pr[b = b'] - \frac{1}{2}
\]

(2)

Definition (2). A T-dPEKS scheme is against an adaptive chosen keyword attack if and only if for any polynomial-time attacker A, its advantage \(Adv_{T-dPEKS,A}^{\text{ind-adv}}(\lambda)\) is negligible.

3.3. Security of trapdoor

The indistinguishability of the trapdoor requires that an adversary cannot learn anything about the keywords to be searched.

Definition (1). Trapdoor Indistinguishability Game.

The game is defined between a challenger B1 and an adversary A1 on security parameter λ and public parameter param. It includes the following stages:

Setup: B1 firstly generates the global parameters param and sends them to the adversary, and then B1 generates the public and private keys for the authorized users and the public key for the designated server pk_s.

Trapdoor queries: A1 can adaptively make the trapdoor queries of keyword w1 for user j, and w2 can be any keyword w2 ∈ {0, 1}λ.

Challenge: A1 chooses between two words w0 and w1, which he/she wants to challenge. B1 chooses a random b’ ∈ {0, 1}λ and computes the trapdoor Tw0 and responds it to A1.

Trapdoor query: A1 can continue to adaptively make the trapdoor queries of w1 for user Uj.

Restriction: w1 should not be w0 or w1.

Output: A1 outputs its guess b ∈ {0, 1}λ for b’, if b = b’, the adversary wins the game.

The advantage of A1 in breaking the trapdoor indistinguishability game in the scheme is defined as follows:

\[
Adv_{T-dPEKS}^{\text{trapdoor}} = \Pr[b = b'] - \frac{1}{2}
\]

(3)

4. Concrete construction

In our scheme, there are three parties: the data owner, the server and the user. When the data owner encrypts his/her data and the corresponding keywords, he/she can also specify the users who can retrieve it, the specific time, and the server which can perform the retrieve at the specific time. Our scheme is composed of the following algorithms:

Setup(λ): Input a security parameter λ, it generates the public parameters param as follows:

(1) Generate an admissible bilinear map e: G1 × G1 → G2, where G1 and G2 are two multiplicative cyclic groups with the same prime order p.

(2) Choose a value x ∈ Z_p∗ randomly, set the master’s private key MSK = x, and then compute the master public key MPK = g^x. Finally, choose the following Hash Functions: H1 : {0, 1}λ → G1, H1 : {0, 1}λ → Z_p, H2 : {0, 1}λ → G1, H3 : G2 → {0, 1}3n.

serverKeyGen(param): Input param, choose ω ∈ Z_p∗ randomly as the server’s private key sks, and set the public key of the server pk_s = [pk_s1, pk_s2] = [g^ω, μ^ω], where g, μ ∈ G1, then the algorithm outputs a pair of public key and private key (pk_s, sk_s) for the server.

userKeyGen(param): Input the identity IDi of user i and the private key x of the master, then the algorithm sets the public key of user i pk_u = H1(IDi), and computes the user’s private key sk_u = x · H1(IDi). Finally, the algorithm outputs a pair of public key and private key (pk_u, sk_u) for user i.

T – dPEKS(param, pk_u, w, pk_s, T): Firstly, input the global parameters param, the public keys pk_u(j = 1, ..., n) of the authorized users, the keyword w, the public key of the designated server pk_s and the specific time T, then choose a number r ∈ Z_p∗ randomly and compute Λ = \(\prod^{n}_{i=1} H1(IDi)\) ∈ Z_p, and then output the keyword ciphertext C = [C1, C2, C3] = [pk_u1, C2, H3(e(H2(w^Λ · H1(T^2) · g^x)))]

Trapdoor(param, sk_u, w, pk_s): Input the global parameter set param, the private key of user i, the public key of pk_s, and the keyword w. Then the algorithm chooses a random value r1 ∈ Z_p∗, and computes the trapdoor Tw corresponding to the keyword w for user i: Tw = [Tw1, Tw2] = [g^r1, H3(e(H2(w^sk_u · H1(pk_s1^r1)))]

Test(C, sk_s, Tw, T): Input ciphertext C, the private key of the designated server sk_s, the time T and the trapdoor Tw. The designated server
uses its private key to compute $T_w = \frac{T_{w2}}{H(T_{w1})} = \frac{H(w)^{\mu_k} \cdot H(p_k^2)}{H(g^{\mu_2})} = H_2(w)^{\mu_k}$, and then tests whether the following equation holds or not:

$$H_1\left(e(T_w, C_1^{(\frac{\mu_2}{\mu_k})\left|g^{\mu_2}\right|})\right) = C_3$$

5. Analysis and comparison

5.1. Correctness proof

Assume that the ciphertext $C = [C_1, C_2, C_3] = [p_k^2, \Delta, H_1(e(H_2(w)^{\Delta} \cdot H_i(T), g)^q)]$ is valid for $w$ and the trapdoor $T_w = [T_{w1}, T_{w2}] = [g^\mu, H_2(w)^{\mu_k} \cdot H(p_k^2)]$ is for $w$. According to the definition, we have

$$T_w = \frac{T_{w2}}{H(T_{w1})} = \frac{H_2(w)^{\mu_k} \cdot H(p_k^2)}{H(g^{\mu_2})} = H_2(w)^{\mu_k}$$

$$= H_2(w)^\mu \cdot H_1(e(T_w, C_1^{(\frac{\mu_2}{\mu_k})\left|g^{\mu_2}\right|}))$$

$$= H_1\left(e(H_2(w)^{\mu} \cdot H_i(T), g)^q\right)$$

$$= H_3(e(H_2(w)^{\Delta} \cdot H_i(T), g)^q)$$

Hence, if $w = w'$, and $H_3(e(H_2(w)^{\Delta} \cdot H_i(T), g)^q) = H_3(e(H_2(w')^{\Delta} \cdot H_i(T), g)^q)$, the algorithm outputs 'yes'.

5.2. Security analysis

In this subsection, we will illustrate the security of our scheme by proving several claims.

5.2.1. Ciphertext indistinguishability

Claim 1. Our scheme is semantically secure against a chosen keyword attack in the random oracle model under the BDH assumption.

Proof. A is supposed to be an adversary who has an advantage $\epsilon$ in breaking the ciphertext indistinguishability of our scheme, and A has at most $q_{ch}$ queries to the hash queries, and $q_{id}$ queries to make trapdoor queries. We design an algorithm B which can violate the BDH problem with a probability at least $\epsilon' = \epsilon/e^{q_{ch}q_{id}}$, where $e$ is the base of the natural logarithm. Since the running time of A and B are approximately the same, so if the BDH assumption holds in $G_1$ and $\epsilon'$ is negligible, then $\epsilon$ must be negligible.

Let g be a generator of $G_1$, algorithm B is given with $g$, and $\mu_1 = g^x$, $\mu_2 = g^y$, $\mu_3 = g^z$ in $G_1$. Its goal is to output $v = e(g, g)^{\rho y} \in G_2$. B simulates the challenger and communicates with A as follows:

Setup: B starts by giving A the master public key $MPK = g^x = \mu_1$ and the public keys of the authorized users $pk_{i_1} = H_1(ID_i)$, where $j = 1, \ldots, n$. Then B forges the server’s public key in this way: B chooses $m, t \in Z_p^*$ randomly and lets the random value $\mu = (\mu_1)^t = g^x$, then it computes the public key of the server $pk_s = [pk_{i_1}, pk_{i_2}, \ldots] = [g^{\mu_1}]m, g^{\mu_2]m, g^{\mu_3} m]$ and there exists an unknown value $sk = x m$. Finally, B sends the public keys of the authorized users and the server to A.

$H_2$ queries: A can make a query to the random oracle $H_2$ arbitrarily. To respond to $H_2$ queries, B maintains a list of tuples $<w_j, h_j, q_j, c_j>$, which is named $H_2 - list$. The list is initially empty. When A queries the random oracle $H_2$ for a keyword $w_j \in \{0, 1\}'$, B responds as follows:

1. If $w_j$ already appears in a tuple $<w_j, h_j, q_j, c_j>$ of $H_2 - list$, then B responds $H_2(w_j) = h_j$ to the query.

2. Otherwise, B generates a random coin $c_i \in \{0, 1\}$, so that Pr[$c_i = 0] = 1/(q_1 + 1)$. Then, B chooses $a_i \in \mathbb{Z}_p$ randomly. If $c_i = 0$, B computes $h_i = \mu_2 \cdot g^z \in G_1$. If $c_i = 1$, B computes $h_i = g^w \in G_1$.

B adds the tuple $<w_j, h_j, q_j, c_j>$ to the $H_2 - list$ and responds to A by setting $H_2(w_j) = h_j$. Note that in both ways, $h_j$ is uniform over $G_1$, so it is independent of A’s view as required.

$H_2$ queries: A can query the random oracle $H_2$ arbitrarily. To respond to $H_2$ queries properly, B needs to set up and maintain a $H_2 - list$ to store the pair $(t, V)$, where

$$t = e(H_2(w)^{\frac{\mu_1}{\mu_2} \cdot H_i(T)})$$

$H_2 - list$ is initially empty. Upon receiving $H_2(t)$ query from A, B responds as follows:

1. If the query for $t$ already appears on the $H_2 - list$ in a tuple $<t, V>$, then B responds with the pair $H_2(t) = V \in \{0, 1\}^n$ to A.

2. Otherwise, B chooses $V \in \{0, 1\}^n$ randomly, and then responds it with $H_2(t) = V$ and stores the pair $(t, V)$ in the $H_2 - list$.

Trapdoor query: When A makes a query for the trapdoor of keyword $w_0$ for user $j$, B replies as follows:

1. B makes $H_2 - query$ to obtain some $h_j \in G_1$, which satisfies $H_2(w_0) = h_j$. Let $<w_j, h_j, q_j, c_j>$ be the corresponding tuple in the $H_2 - list$. If $c_j = 0$, B halts.

2. Otherwise, as $c_j = 1$, hence $h_j = g^w \in G_1$, then B chooses $r \in \mathbb{Z}_p$ randomly and sets $T_w = [g^r, H_1(H_i(ID_i)), H_2(w)^{\mu_2}]$. B responds $T_w$ to A. Note that $\mu_2^H \cdot (H_2(w)^{\mu_2}) \in G_1$, so $T_w$ is independent of A’s as required.

Challenge: First A generates a pair of keywords $w_0$ and $w_1$ to be challenged, then B executes the challenge as follows:

1. B makes $H_2$ queries twice to obtain $h_0, h_1 \in G_1$, such that $H_2(w_0) = h_0$ and $H_2(w_1) = h_1$. As $i \in \{0, 1\}$, let $<w_i, h_i, q_i, c_i>$ be the corresponding tuples in the $H_2 - list$. If both $c_i = 0$ and $c_j = 1$, B terminates.

2. Otherwise, since at least one of $c_0, c_1$ equals 0, B picks a $b \in \{0, 1\}$ randomly, such that $c_b = 0$.

3. B responds to the challenge as follows: B chooses $k \in \mathbb{Z}_p'$ randomly, then sets $C_1^i = pk_{i_1}^k = (g^x)^{\frac{m}{\mu_1} b}$ and responds $C_1 = [C_1, \Delta, J] \rightarrow A$ for a random $J \in \{0, 1\}$. Actually this challenge implicitly defines $J = H_2(e(H_2(w)^{\mu_2} \cdot H_i(ID_i), H_i(T)) \cdot \mu_1^2))$. Since $c_1 = 0, H_2(w) = \mu_2 \cdot g^x$, we have

$$J = H_1\left(e(H_2(w)^{\mu_2} \cdot H_i(ID_i), H_i(T) \cdot \mu_1^2)\right)$$

$$= H_1\left(e(\mu_1 \cdot g^x \cdot g^k \cdot (g^y)^{\frac{m}{\mu_1} b} \cdot H_i(T))\right)$$

$$= H_1\left(e(g^x, g^y, g^k \cdot (g^y)^{\frac{m}{\mu_1} b} \cdot H_i(T))\right)$$

According to the definition, $C_1^*$ is a valid ciphertext for $w_0$.

More trapdoor queries: A can issue trapdoor queries for a keyword $w_0$ once again, with the only constraint being $w_0 \neq w_0, w_1$. B replies to the queries as before.
Output: Finally, A outputs its guess $b' \in \{0, 1\}$, which implies the challenge ciphertext $C$, which is $T - \text{dpeks}(\text{param}, pk_A, w_0, pk_T)$ or $T - \text{dpeks}(\text{param}, pk_A, w_1, pk_T)$. Then B chooses a pair $(t, v)$ randomly from $H_3 - \text{list}$ and outputs $\frac{\text{H}_0(e(g, g)^{y v})}{e(g, g)^{y t}} = e(g, g)^{y v}$ as its guess for $e(g, g)^{y v}$, where $\epsilon_0$ is the value appeared in the challenge step.

In fact, A must issue a query for either $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ or $H_2(e(H_2(w_1)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$. And this means that the $H_3 - \text{list}$ contains a pair in which $t = e(H_2(w_1)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i) = e(g, g)^{y v}$. If B can find this pair $(t, v)$ from the $H_3 - \text{list}$, then $\frac{\text{H}_0(e(g, g)^{y v})}{e(g, g)^{y t}} = e(g, g)^{y v}$ is required.

Next, we will analyze the probability of which B does not halt during the games. For convenience, we define three events as follows:

$\epsilon_1$: B does not abort during any trapdoor queries.
$\epsilon_2$: B does not abort during the challenge step.
$\epsilon_3$: A does not make a query for either $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ or $H_2(e(H_2(w_1)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$.

Next, we will illustrate the non-negligible probability that events $\epsilon_1$ and $\epsilon_2$ occur.

Claim 2. The probability that B does not abort during any trapdoor queries is at least $1/e$. That is: $\text{Pr}[\epsilon_1] \geq 1/e$.

Proof. Not to lose generality, we suppose that A will not query the trapdoor for the same keyword twice. Let $w_i$ be the $ith$ trapdoor query for A and $\langle w_i, H_i, a_i, c_i \rangle$ be the associated entry in the $H_2 - \text{list}$. By the previously given definitions, it is easy to find that $c_i$ is independent of A’s view. However, no matter $c_i = 0$ or $c_i = 1$, the distribution on $H_2(w_i)$ is the identical. Thus the query can be halting with at least $1/(q_T + 1)$. Since A can make at most $q_T$ trapdoor queries, so the probability that B does not halt during any A’s trapdoor queries is at least $1 - (1/(q_T + 1))^q_T \geq 1/e$.

Claim 3. The probability that B does not abort during the challenge steps is at least $1/q_T$. That is: $\text{Pr}[\epsilon_2] \geq 1/q_T$.

Proof. Based on the above game definition, we know that B will halt at the challenge step if A can generate $w_0, w_1$, which satisfy the following qualifications: $c_0 = c_1 = 1$, for $i = 0, 1$, and $w_i, H_i, a_i, c_i >$ is the corresponding tuple in the $H_2 - \text{list}$. As A hasn’t queried the trapdoor for $w_0, w_1$, both $c_0$ and $c_1$ is independent of A’s view. Therefore, for $i = 0$ and $i = 1$, $\text{Pr}[c_0 = 0] = 1/(q_T + 1)$, and those two values are independent of each other. We have $\text{Pr}[c_0 = c_1 = 1] = (1 - 1/(q_T + 1))^2 \leq 1 - 1/q_T$. Thus, the probability that B will not abort is at least $1/q_T$.

Based on the definition as mentioned above, it is easy to know that A is not allowed to issue a trapdoor query on $w_0$ and $w_1$, and events $\epsilon_1$ and $\epsilon_2$ are independent of each other, so $\text{Pr}[\epsilon_1 \land \epsilon_2] \geq 1/eq_T$.

Next we will demonstrate that B can output the solution for the mentioned BDH instance with the probability being at least $\epsilon q_T$. To achieve this, we will demonstrate that A is able to issue a query for $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ with the probability being at least $\epsilon q_T$ during the game.

Claim 4. Suppose in a real attacking game, A is given an authorized user’s public keys $pk_A$ and A is going to challenge on keywords $w_0$ and $w_1$. In the response, A is replied a challenge $C = [g', J]$. Then, in the real attack game, the probability of A to initiate an $H_i$ query for either $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ or $H_2(e(H_2(w_1)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ is at least $2\epsilon q_T$.

Proof. If $\epsilon_3$ occurs, we know the bit $b \in \{0, 1\}$, which indicates that whether C is the ciphertext of $w_0$ or $w_1$ is independent of A’s view. Hence, the probability that A outputs the appropriate bit $b = b$ is at least $\frac{1}{2}$. Based on the definition as mentioned before, it is easy to find that $1/e \geq \epsilon_3$, When $\text{Pr}[\epsilon_3] \geq 2\epsilon q_T$, the illustration is as follows:

$\text{Pr}[b = b'] = \text{Pr}[b = b'|s_i] \cdot \text{Pr}[s_i] + \text{Pr}[b = b'|s_i] \cdot \text{Pr}[s_i] \leq \text{Pr}[b = b']$.

$\epsilon_3 = 1 - 2\epsilon q_T \leq \frac{1}{2} \cdot \text{Pr}[s_i] + \text{Pr}[s_i] = \frac{1}{2} \cdot \text{Pr}[s_i]$.

So we have $\epsilon \leq \frac{1}{2} \cdot \text{Pr}[b = b']$.

Assume B does not halt, then the simulation in a real attack game for B can last up to the moment when A issues a query for $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ or $H_2(e(H_2(w_1)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$. Based on Claim 4, the probability that A will make a query for $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ or $H_2(e(H_2(w_1)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ is at least $2\epsilon q_T$. Hence, the probability that A issues a query for $H_2(e(H_2(w_0)^{y t}, H_i(\Pi)) \cdot H(T), \mu'_i)$ is at least $\epsilon$.

As we know, the probability that the value appear on the left side of a pair in $H_3 - \text{list}$ is identical, so B is able to choose the right pair with the probability being at least $1/q_T$. Suppose B does not halt during the simulation game, B can produce the correct answer with the probability being at least $1/q_T$. As the probability that B does not halt is at least $1/q_T$, the probability of B’s success is at least $\epsilon q_T$.

5.2.2. Trapdoor indistinguishability

Claim 5. The presented scheme satisfies the trapdoor indistinguishability under the HDH assumption.

Proof. Assume there exists an adversary $A_1$, with an advantage $\epsilon$ in breaking the trapdoor indistinguishability of the proposed scheme, and $A_1$ can make at most $q_T$ trapdoor queries, where $q_T$ is a positive. Then, we can construct an algorithm $B_1$ which has an advantage $\epsilon' - \epsilon$ to solve the HDH problem in $G_1$. $B_1$ inputs a $(g, g^a, g^b, \eta) \in G_1 \times \mathbb{Z}$ and $H : \{0, 1\}^* \rightarrow G_1$, where $\eta$ is whether $H(g^b)$ or a random element of $G_1$. $B_1$ acts as the challenger and interacts with $A_1$ as follows:

Setup: $B_1$ runs Setup($\lambda$) to generate the global parameters param, and sets the user’s public key $pk_B = H_1(D_1)$ and private key $sk_A = x \cdot H_1(D_1)$. $B_1$ chooses $a, b, \lambda \in Z_p$ randomly, sets $\mu = (g^a)^{\lambda}$ and the server’s public key $pk = \langle pk_A, pk_B \rangle = ((g^a)^{\lambda})^{\lambda}, (g^b)^{\lambda})$, and there exists an unknown value $\theta$ according to the definition $sk_A = \theta = \alpha$.

Trapdoor query: When $A_1$ issues a query for a word $w_i$ for user $j$, $B_1$ responds as follows: $B_1$ chooses a random value $\tau' \in Z_p$, sets $T_1' = g^\tau$, $T_2' = H_2(w_i)^{x \cdot H(D_1)} \cdot H(g^{b'})$, where $\tau \in Z_p$ is the value selected in the setup step, and then $B_1$ responds to $A_1$ with $T_1' = [T_1', T_2']$.

Challenge: $A_1$ chooses between two keywords $w_0, w_1$, which he/she wishes to challenge. $B_1$ chooses $b' \in \{0, 1\}$ randomly, and generates $T_0$ as follows:

(1) Sets $T_1' = (g^\lambda)^{\lambda}$ and $T_2' = H_2(w_i)^{y \cdot H(D_1)} \cdot \eta$, which $\eta$ and $g^b$ are components of HDH instance. In fact this challenge defines $T_2' = H_2(w_i)^{y \cdot H(D_1)} \cdot H(g^{ab})$, $H(g^{ab})$ is uniform in $G$ and $T_0$ is a valid trapdoor of $w_i$, so in the adversary’s view, it is independent of the bit $b'$.
Table 1
The notations.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ct-Ind</td>
<td>Ciphertext indistinguishability</td>
</tr>
<tr>
<td>Tr-Ind</td>
<td>Trapdoor indistinguishability</td>
</tr>
<tr>
<td>Tm-RE</td>
<td>Timed-release</td>
</tr>
<tr>
<td>De-S</td>
<td>Designated server</td>
</tr>
<tr>
<td>Size(Trap)</td>
<td>The size of the trapdoor</td>
</tr>
<tr>
<td>Comp(Match)</td>
<td>Computational cost of the Test stage</td>
</tr>
<tr>
<td></td>
<td>The size of element in G</td>
</tr>
<tr>
<td>Tp</td>
<td>Bilinear paring operation</td>
</tr>
<tr>
<td>Te</td>
<td>Exponentiation operation</td>
</tr>
<tr>
<td>Enc</td>
<td>Encryption operation</td>
</tr>
<tr>
<td>Dec</td>
<td>Decryption operation</td>
</tr>
</tbody>
</table>

Table 2
Comparison of security and performance.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ct-Ind</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Tr-Ind</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Tm-RE</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Size(Trap)</td>
<td>2(G)</td>
<td></td>
<td>G</td>
<td></td>
<td>2(G)</td>
</tr>
<tr>
<td>Comp(Match)</td>
<td>Tp + Tp</td>
<td>Tp + Dec</td>
<td>Tp + 2Tp</td>
<td>2T + 3T + Tp</td>
<td>Tp + Tp</td>
</tr>
</tbody>
</table>

**Trapdoor query:** A1 can issue a trapdoor query for keyword w1. B1 responds in the way as mentioned above.

**Restriction:** The only restriction is that w1 ≠ w0, w2.

**Output:** A1 outputs its guess b ∈ {0, 1}. If b = b1, then B1 outputs 1, which means η = H(gab), and outputs 0 otherwise.

**Probability analysis:** We will demonstrate that when the input tuple is a sample from HDH (where η = H(gab)), since H(gab), \[Pr[b = b1] = \frac{1}{2} + \frac{1}{2} > \epsilon\] holds. On the other hand, when the input tuple is sampled from HDH (where η is uniform over G1), then η and T2 = H2(w2) are uniform and independent over G1 in which case \[Pr[b = b1] = \frac{1}{2}\]. Therefore, with g, g4, g6, and η being uniform over G1, we have \[Pr[B1(g, g4, g6, H(gab))] = 0 - Pr[B1(g, g4, g6)] = 0\].

In this simulation, B1 does not abort at any step, hence we can know that the overall probability of success for B1 is the same as that of A1.

5.3. Comparison

To evaluate the security and performance of the proposed scheme, we compare our scheme with some similar schemes. As the proposed scheme is distributed architecture oriented, the communication cost is a critical factor for the evaluation, so we compare the sizes of the trapdoors which will be transferred via the network frequently. For a better search experience, the cost of the match operation performed by the designated server should be as cheap as possible, so we concentrate on the comparison of match operation. For simplicity, some notations are defined in Table 1, and the overall comparison is given in Table 2.

As shown in Table 2, Rhee et al.'s scheme needs an exponentiation operation and a bilinear paring operation [12]; Boneh et al.'s scheme needs an exponentiation operation and a decryption operation [18]; Wu et al.'s scheme needs an exponentiation operation and two bilinear paring operations [14]; Ibraimi et al.'s scheme needs two exponentiation operations and three bilinear paring operations [7]; our scheme needs an exponentiation operation and a bilinear paring operation. From the above comparison, we can see that the computational cost of our scheme is the same as Rhee et al.'s [12] and less than those of other schemes [7, 14]. Although the computational cost of Boneh et al.'s scheme is less than that of ours, their scheme can not achieve the effect of trapdoor indistinguishability and designated server. Furthermore, only our scheme has the timed-release feature. Therefore, our scheme is more practical for EHRs.

6. Conclusion

In this paper we propose a time-aware dPEKS scheme for EHRs. Compared with previous dPEKS schemes, this scheme not only has the properties of the timed-release cryptography but also achieves a control over the user’s access. We also demonstrate that our scheme possesses the ciphertext indistinguishability and trapdoor indistinguishability under the BDH and the HDH assumptions.

Acknowledgements

This study was jointly supported by the National Natural Science Foundation of China (No. 61702067, No. 61472464), the Natural Science Foundation of Shandong Province, China (No. ZR2015FL024).

References


