Searchable Encryption with randomized ciphertext and randomized keyword search

Marco Calderini*, Riccardo Longo*, Massimiliano Sala*, Irene Villa*

*Department of Mathematics, University of Trento

Abstract

The notion of public key encryption with keyword search (PEKS) was introduced to efficiently search over encrypted data. In this paper, we propose a PEKS scheme in which both the encrypted keyword and the trapdoor are randomized, so that the cloud server is not able to recognize identical queries. Our scheme is CI-secure in the single user setting and TI- secure in the multi-user setting with multi-trapdoor.

1 Introduction

With the rapid development of cloud computing technology, more and more enterprises and individuals are willing to share their own data on cloud platforms.

Since the data owner loses its control of the data and cloud servers may be untrusted, several security and privacy issues arise in cloud storage. So, sensitive data should be encrypted before uploading to cloud server, to protect the data from being leaked. However, data encryption makes it extremely difficult to search for a specific file in a large number of encrypted files.

In the last years, many prominent cryptographic primitives have been proposed for achieving secure and efficient cloud-data usage, such as searchable encryption (SE) [2, 18]. Searchable encryption allows a remote server to search in the encrypted data on behalf of a client without the knowledge of plaintext data.

Almost all SE techniques provide search ability over encrypted documents by extracting the keywords of those documents and generating searchable ciphertexts corresponding to these keywords [6, 7, 14, 15, 17]. Then, data receivers can search uploaded encrypted documents to find those containing a keyword by generating a trapdoor to send to the server. Once received the trapdoor, the server runs an algorithm to test which documents contains the searched keyword. If there is a match, then the server returns the associated encrypted document.

The first SE schemes that appeared in the literature use a symmetric setting [18].

In 2004, Boneh et al. [2] proposed the first public key encryption with keyword search (PEKS). In a multi-user setting, PEKS ([2]) allows any user to encrypt keywords for searching by designated searching key holders. However, PEKS schemes are vulnerable to off-line Keyword Guessing Attacks (KGA) [5, 9]. That is, given a trapdoor, the adversary can generate a ciphertext of a guessing keyword and then test whether it matches with a the trapdoor. If the keyword space has low entropy, this attack will be very efficient. Indeed, several PEKS schemes are shown to be insecure against KGAs [5, 9, 11, 14, 19, 20].

Public-key Authenticated Encryption with Keyword Search (PAEKS) has been proposed by Huang and Li [8], in 2017, to defend against keyword guessing attacks. Its security model guarantees two security goals: cipher-keyword indistinguishability (CI-security) and trapdoor indistinguishability (TI-security). However, recently, Noroozi et al. [13] showed that the PAEKS scheme in [8] is not secure in the multi-user setting, and Qin et al. [15] showed that it is not secure in the multi-cipher-keyword setting. Both [13, 15] proposed some adjustment to the scheme.

Qin et al., in [16], proposed a PAEKS scheme and they prove that their scheme is secure in the multi-cipher-keyword setting for CI-security and in the multi-user setting. However, in [16], the trapdoor is deterministic, thus, if an attacker is allowed to issue a trapdoor query for any challenge keyword, then the scheme is not secure in a multi-trapdoor-keyword setting.

In this paper, we propose a PAEKS scheme with an improved TI-security model with respect to [16]. In particular, our PAEKS scheme is CI-secure in the single-user scenario and TI-secure in the multi-trapdoor and multi-user scenario. Moreover, both the encryption and the trapdoor are randomized, that is, encrypting two times the same keyword (or creating the trapdoor for the same keyword) will produce different results, differently from [16] where the trapdoor is not randomized. Note that, if we exchange the role of the of the encryption and trapdoor algorithm, we would obtain a PAEKS scheme CI-secure in the multi-user setting and TI-secure in the single user setting.

This paper is structured as follows. In Section 2 we provide some preliminary notions, useful to understand the rest of the paper. Section 3 presents PEKS and PAEKS schemes, together with the security models for trapdoor indistinguishability and ciphertext indistinguishability. In Section 4 we describe our PAEKS scheme, commenting on the possibility of combining multiple keywords, and in Section 5 we provide the security proofs of our scheme. In particular, we prove that our PAEKS scheme is fully secure in a multi-user setting for trapdoor indistinguishability, and it is secure in a single-user setting for ciphertext indistinguishability. The conclusions of our work are in Section 6.

2 Preliminaries

In this section, we collect the notations and preliminaries needed for the rest of this work. The symbol $\mathbb{Z}$ stands for the ring of integers and, for n a positive integer, $\mathbb{Z}_n$ is the ring of integers modulo n.

2.1 Bilinear pairing

Let G, G_T be two multiplicative cyclic groups of prime order p. An admissible pairing is a map $e : G \times G \to G_T$ satisfying the following properties.

- **Bilinearity**: for any $g, h \in G$ and $a, b \in \mathbb{Z}$, $e(g^a, h^b) = e(g, h)^{ab}$.
- **Non-degeneracy**: for any generator g of G, $e(g, g) \in G_T$ is a generator of G_T.
- **Computability**: for any $g, h \in G$, we can compute $e(g, h)$ efficiently.

Bilinear pairings play an important role in the construction of many cryptographic schemes such as Identity-Based Encryption schemes [3], Attribute-Based Encryption schemes [1], Key-Agreement protocols [10], Signature schemes [4], etc. Many schemes based on pairings can be found in a recent survey on Functional Encryption in [12].

2.2 Complexity Assumptions

We recall some problems that are believed to be hard. The related assumptions will be used in the proof of security of the proposed scheme.

A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for any positive integer d there exists an integer N_d such that $|f(k)| < \frac{1}{k^d}$ for any $k \geq N_d$.

We define the advantage of an algorithm $\mathcal{A}$ that outputs a guess β' of a bit β as $\mathrm{Adv}_{\mathcal{A}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 1 (CDH)**.** *The Computational Diffie-Hellman (CDH) Problem over a group G of order p is the following. Given a generator $g \in G$ and two elements $g^x, g^y \in G$, for x, y randomly chosen from $\mathbb{Z}_p$, compute the element g^{xy}. We say that CDH is intractable (i.e. the CDH assumption holds) if all polynomial time algorithms have a negligible probability of solving CDH.*

Notice that, when we are considering an admissible bilinear map, solving the decisional version of the above problem is easy.

Definition 2 (DBDH)**.** *The Decisional Bilinear Diffie–Hellman (DBDH) Problem over a bilinear pairing (G, G_T, e) of order p is the following. Given a generator $g \in G$ and elements $g^x, g^y, g^z \in G$ where x, y, z are randomly chosen from $\mathbb{Z}_p$, distinguish $e(g, g)^{xyz}$ from a random element of G_T. We say that DBDH is intractable (i.e. the DBDH assumption holds) if all polynomial time algorithms have a negligible advantage in solving DBDH.*

If we assume that the DBDH is intractable, then also the CDH is intractable.

Definition 3 (DLIN)**.** *The Decisional Linear (DLIN) Problem over a group G of order p is the following. Given a generator $g \in G$ and the elements $g^x, g^y, x^{xr}, g^{ys} \in G$, for x, y, s, r randomly chosen from $\mathbb{Z}_p$, distinguish g^{r+s} from a random element of G.*

As in [8], we consider the modified Decisional Linear (mDLIN) problem, which is defined as below.

Definition 4 (mDLIN)**.** *Given a generator $g \in G$ and the elements $g^x, g^y, g^{xr}, g^{s/y} \in G$, for x, y, s, r randomly chosen from $\mathbb{Z}_p$, distinguish g^{r+s} from a random element of G. We say that mDLIN is intractable (i.e. the mDLIN assumption holds) if all polynomial time algorithms have a negligible advantage in solving mDLIN.*

Similarly as before, if we assume that the mDLIN is intractable, then also the CDH is intractable.

3 Preliminaries on public-key searchable encryption schemes

In this section, we introduce PEKS and PAEKS schemes and the related security notions.

3.1 Public-key encryption with keyword search

A public-key encryption with keyword search (PEKS) consists of the following (probabilistic) polynomial-time algorithms [2].

- `Setup`(λ): given in input a security parameter λ, the algorithm outputs a global system parameter Param.
- `KeyGen`(Param): given the system parameter, it outputs a pair of public and secret keys (pk, sk). The algorithm is run by the data receiver.
- `Encrypt`(W, pk): given a keyword W and the receiver's public key, it outputs a ciphertext C_W of W. The algorithm is run by the data sender.
- `Trapdoor`(W, sk): given a keyword W and the secret key, it outputs a trapdoor T_W. The algorithm is run by the data receiver.
- `Test`(pk, $C_{W'}$, T_W): given the receiver's public key, a ciphertext $C_{W'}$ and a trapdoor T_W, it outputs 1 (`true`) indicating that $C_{W'}$ and T_W contain the same keyword ($W' = W$), and 0 otherwise. The algorithm is run by the cloud server.

The first bilinear pairing-based PEKS scheme was proposed in 2004 [2].

This type of scheme is vulnerable to the inside keyword guessing attack. Indeed, to recover the keyword contained in a trapdoor T_W, a honest-but-curious cloud server could check whether W' equals to the keyword W contained in T_W by computing the ciphertext $C_{W'}$ and performing the test algorithm. Since in real applications the keyword space is usually not that big, the server would be able to finish the keyword guessing attack in a reasonably short time.

To address to this issue, the notion of publick-key authenticated encryption with keyword search was introduced in 2017 [8].

3.2 Public-key authenticated encryption with keyword search

A public-key authenticated encryption with keyword search (PAEKS) scheme consists of the following (probabilistic) polynomial-time algorithms.

- `Setup`(λ): given in input a security parameter λ, the algorithm outputs a global system parameter Param.
- $\texttt{KeyGen}_S$(Param): given the system parameter, the sender's key pair generation algorithm outputs a pair of public and secret keys $(\mathrm{pk}_S, \mathrm{sk}_S)$ for the sender.
- $\texttt{KeyGen}_R$(Param): given the system parameter, the receiver's key pair generation algorithm outputs a pair of public and secret keys $(\mathrm{pk}_R, \mathrm{sk}_R)$ for the receiver.
- $\texttt{Encrypt}(W, \mathrm{sk}_S, \mathrm{pk}_R)$: given a keyword W, the receiver's public key and the sender's private key, it outputs a ciphertext C_W of W. The algorithm is run by the data sender.
- $\texttt{Trapdoor}(W, \mathrm{pk}_S, \mathrm{sk}_R)$: given a keyword W, the sender's public key and the receiver's secret key, it outputs a trapdoor T_W. The algorithm is run by the data receiver.
- $\texttt{Test}(\mathrm{pk}_S, \mathrm{pk}_R, C_{W'}, T_W)$: given the receiver's public key, the receiver's public key, a ciphertext $C_{W'}$ and a trapdoor T_W, it outputs 1 (`true`) indicating that $C_{W'}$ and T_W contain the same keyword, and 0 otherwise. The algorithm is run by the cloud server.

As explained in [8], the notion of PAEKS prevents a third-party, even the cloud server, from generating a valid ciphertext-keyword. It provides both confidentiality and integrity of the plaintext.

3.3 Security models

Similar with PEKS, security of PAEKS requires that there is no probabilistic polynomial-time adversary which could distinguish trapdoors or ciphertexts. Therefore, a semantic security model for PAEKS includes both ciphertext indistinguishability (CI-security) and trapdoor indistinguishability (TI-security or Trapdoor Privacy). Related to the security of PAEKS schemes, we recall the notation presented in [16], declined to the purposes of our scheme. Suppose that $(\mathrm{pk}_S, \mathrm{sk}_S)$ and $(\mathrm{pk}_R, \mathrm{sk}_R)$ are the key pairs of the attacked data sender and data receiver respectively. In a multi-user settings, an adversary may have the following two abilities to attack a PAEKS scheme.

Chosen keyword to ciphertext (CKC) attacks In a CKC attack, the adversary has the ability to obtain a ciphertext for any keyword W of its choice under a receiver's public key $\overline{\mathrm{pk}}_R$ specified by the adversary. That is, the adversary will obtain the ciphertext $C_W = \texttt{Encrypt}(W, \mathrm{sk}_S, \overline{\mathrm{pk}}_R)$. Formally, CKC attacks are modelled by giving the adversary $\mathcal{A}$ access to a ciphertext oracle $\texttt{Encrypt}_{\mathrm{sk}_S}(\cdot,\cdot)$, viewed as a "black box"; the adversary can repeatedly submit any keyword W and a (data receiver's) public key $\overline{\mathrm{pk}}_R$ of its choice to this oracle, and is given in return a ciphertext $C_W = \texttt{Encrypt}(W, \mathrm{sk}_S, \overline{\mathrm{pk}}_R)$.

Chosen keyword to trapdoor (CKT) attacks In a CKT attack, the adversary has the ability to obtain a trapdoor of any keyword W of its choice under a sender's public key $\overline{\mathrm{pk}}_S$ specified by the adversary. That is, the adversary will obtain the trapdoor $T_W = \texttt{Trapdoor}(W, \overline{\mathrm{pk}}_S, \mathrm{sk}_R)$. Similarly, CKT attacks are modelled by giving the adversary $\mathcal{A}$ access to a trapdoor oracle $\texttt{Trapdoor}_{\mathrm{sk}_R}(\cdot,\cdot)$, viewed as a "black box"; the adversary can repeatedly submit any keyword W and a (data sender's) public key $\overline{\mathrm{pk}}_S$ of its choice to this oracle, and is given in return a trapdoor $T_W = \texttt{Trapdoor}(W, \overline{\mathrm{pk}}_S, \mathrm{sk}_R)$.

Clearly, the adversary's access to the above-mentioned oracles has to be restricted to some trivial instances. Let W_0^* and W_1^* be the challenge keywords, then, in the CI-security model, the adversary cannot request trapdoors of the challenge keywords for the target public keys, lest the challenge become trivial. In many PAEKS schemes, such as [8], there are other limitations on the ciphertext oracle. For example, the adversary is not allowed to request the ciphertext corresponding to either W_0^* or W_1^*. In [16], they consider *full CKC attacks*, where this limitation is removed.

Similarly, for TI-security, the adversary cannot request ciphertext of the challenge keywords. Also in this case, many PAEKS schemes (see [8, 16]) impose other limitations such as the adversary is not allowed to request trapdoors corresponding to either W_0^* or W_1^*. In this paper we consider *full CKT attacks*, where this limitation is removed.

In the following, we present the formal definitions of the security notions we will use.

3.3.1 (MU) Full TI-Security Model

We describe the *Full TI-Security Game* for an adversary $\mathcal{A}$ in the multi-user setting.

1. Initializationn: Given a security parameter λ, the challenger runs the setup algorithm to generate the global system parameter Param. Then, it runs $\texttt{KeyGen}_S(\mathrm{Param})$ and $\texttt{KeyGen}_R(\mathrm{Param})$ to generate the target sender's key pair $(\mathrm{pk}_S, \mathrm{sk}_S)$ and the target receiver's key pair $(\mathrm{pk}_R, \mathrm{sk}_R)$ respectively. The challenger invokes the adversary $\mathcal{A}$ on input $(\mathrm{Param}, \mathrm{pk}_S, \mathrm{pk}_R)$.

2. Phase 1: The adversary is allowed to adaptively issue queries to the following oracles for polynomially many times.

    - **Trapdoor Oracle $\mathcal{O}_T$:** Given a keyword W and a public key $\overline{\mathrm{pk}}_S$, the oracle computes the corresponding trapdoor T_W with respect to $\overline{\mathrm{pk}}_S$ and sk_R, and returns T_W to $\mathcal{A}$.
    - **Ciphertext Oracle $\mathcal{O}_C$:** Given a keyword W and a public key $\overline{\mathrm{pk}}_R$, the oracle computes the corresponding ciphertext C_W with respect to sk_S and $\overline{\mathrm{pk}}_R$, and returns C_W to $\mathcal{A}$.

3. Challenge: At some point, $\mathcal{A}$ chooses two keywords (W_0^*, W_1^*) with the restriction that (W_0^*, pk_R) and (W_1^*, pk_R) have never been queried to $\mathcal{O}_C$ in Phase 1. These keywords are submitted to the challenger as the challenge keywords. The challenger randomly chooses a bit $\beta \in \{0, 1\}$, computes $T_{W_\beta^*} \leftarrow \texttt{Trapdoor}(W_\beta^*, \mathrm{pk}_S, \mathrm{sk}_R)$ and returns $T_{W_\beta^*}$ to $\mathcal{A}$.

4. Phase 2: The adversary continues to issue queries to $\mathcal{O}_T$ and $\mathcal{O}_C$ as above, with the restriction that neither (W_0^*, pk_R) nor (W_1^*, pk_R) could be submitted to the ciphertext oracle.

5. Guess: Finally, $\mathcal{A}$ outputs a bit $\beta' \in \{0, 1\}$. It wins the game if and only if $\beta' = \beta$.

We define $\mathcal{A}$'s advantage of successfully distinguishing the trapdoors of the scheme as $\mathrm{Adv}_{\mathcal{A}}^{T}(\lambda) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 5 ((MU) Full TI-security)**.** *A PAEKS scheme satisfies trapdoor indistinguishability under a full CKT attack and a CKC attack in the multi-user setting, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage $\mathrm{Adv}_{\mathcal{A}}^{T}(\lambda)$ is negligible in λ.*

3.3.2 CI-Security Model

We describe the *CI-Security Game* for an adversary $\mathcal{A}$, in the single-user scenario.

1. Initializationn: Given a security parameter λ, the challenger generates Param and prepares $\mathrm{pk}_S, \mathrm{pk}_R$ as in the previous Game. It then invokes the adversary $\mathcal{A}$ on input $(\mathrm{Param}, \mathrm{pk}_S, \mathrm{pk}_R)$.

2. Phase 1: The adversary issues queries to oracles $\mathcal{O}_Q$ and $\mathcal{O}_C$ as before, but declined to a single-user setting (no public key is given in input to the oracles).

3. Challenge: At some point, $\mathcal{A}$ chooses two keywords (W_0^*, W_1^*) which have not been requested for trapdoors nor ciphertexts, and submits them to the challenger as the challenge keywords. The challenger randomly chooses a bit $\beta \in \{0, 1\}$, computes $C_{W_\beta^*} \leftarrow \mathrm{Encrypt}(W_\beta^*, \mathrm{sk}_S, \mathrm{pk}_R)$ and returns $C_{W_\beta^*}$ to $\mathcal{A}$.

4. Phase 2: The adversary continues to issue queries to $\mathcal{O}_Q$ and $\mathcal{O}_C$ as above, with the restriction that neither W_0^* nor W_1^* could be submitted to either oracle.

5. Guess: Finally, $\mathcal{A}$ outputs a bit $\beta' \in \{0,1\}$. It wins the game if and only if $\beta' = \beta$.

We define $\mathcal{A}$'s advantage of successfully distinguishing the ciphertexts of our scheme as $\mathrm{Adv}_{\mathcal{A}}^{C}(\lambda) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 6 (CI-Security)**.** *A PAEKS scheme satisfies ciphertext indistinguishability under a CKT attack and a CKC attack, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage $\mathrm{Adv}_{\mathcal{A}}^{C}(\lambda)$ is negligible in λ.*

4 The new scheme

In this section, we describe a new public-key authenticated searchable encryption scheme. For the sake of brevity, we condense the two algorithms $\texttt{KeyGen}_S$ and $\texttt{KeyGen}_R$ into a single step `KeyGen`.

- `Setup`. Given a security parameter λ, the algorithm constructs two multiplicative groups of prime order p, G and G_T, a random generator g of group G and an admissible bilinear map $e : G \times G \to G_T$. Then, it selects two hash functions $H : \{0,1\}^* \to G$ and $H_2 : G \to \mathbb{Z}_p^*$. The global system parameters are $\mathrm{Param} = (G, G_T, e, p, g, H, H_2)$.

- `KeyGen`. Given Param, the algorithm produces the following pair of keys for sender and receiver: $(\mathrm{pk}_S, \mathrm{sk}_S) = (g^a, a)$ and $(\mathrm{pk}_R, \mathrm{sk}_R) = (g^b, b)$, with $a, b \in \mathbb{Z}_p^*$ chosen randomly. From them, the sender and the receiver can construct three common secrets: $h, t \in G$ and $s \in \mathbb{Z}_p^*$. In particular, $h = g^{ab}$, $t = g^{H_2(h)}$ and $s = H_2(t)$.

- `Encrypt`. Given a keyword $W \in \{0,1\}^*$, pk_R and sk_S, the common secrets h, t, s are obtained. The sender selects a random $r \in \mathbb{Z}_p^*$ and outputs the pair
$$C_W = \big[C_1, C_2\big] = \big[t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W)^s \cdot g^r, h^r\big].$$

- `Trapdoor`. Given a keyword $W \in \{0,1\}^*$, pk_S and sk_R, the common secrets h, t, s are obtained. The receiver selects a random $\rho \in \mathbb{Z}_p$ and outputs the tuple
$$T_W = \big[Q_1, Q_2, Q_3\big] = \big[e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W)^s, h^\rho), g^\rho, h^\rho\big].$$

- `Test`. Given in input any trapdoor $T_{W'} = \big[Q_1, Q_2, Q_3\big]$, corresponding to a keyword W', and any ciphertext $C_W = \big[C_1, C_2\big]$, corresponding to a keyword W, the test consists in checking the equivalence
$$e(C_1, Q_3) = Q_1 \cdot e(C_2, Q_2).$$

The correctness of the scheme is verified as follows. Since

$$
\begin{aligned}
e(C_1, Q_3) =& e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W)^s \cdot g^r, h^\rho) \\
=& e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W)^s, h^\rho) \cdot e(g^r, h^\rho), \\
Q_1 \cdot e(C_2, Q_2) =& e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W')^s, h^\rho) e(h^r, g^\rho) \\
=& e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W')^s, h^\rho) e(g^r, h^\rho),
\end{aligned}
$$

if the keywords W and W' are the same, then $H(\mathrm{pk}_S||\mathrm{pk}_R||W) = H(\mathrm{pk}_S||\mathrm{pk}_R||W')$ and the equivalence is satisfied. If the keywords are different ($W \neq W'$), then $H(\mathrm{pk}_S||\mathrm{pk}_R||W) \neq H(\mathrm{pk}_S||\mathrm{pk}_R||W')$ due to collision resistance of the hash function H, and thus $Q_1 \neq e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W)^s \cdot g^r, h^\rho)$.

Remark 1. *The aim of the* `KeyGen` *algorithm is to generate three secrets only known by the sender and the receiver. Notice that there are also other options in order to construct these secrets.*

4.1 On the combination of keywords

Suppose that we want to allow the search also for combinations of keywords. That is, given two distinct keywords, we want to allow the search of documents containing the first keyword, the second keyword or both keywords.

A trivial way to do this is, for the sender, to generate a ciphertext for each keyword characterizing the document. The receiver, who wants to find a document containing n different keywords $W_1, \ldots, W_n$, will have to verify that, among these ciphertexts, the test algorithm is satisfied at least once for all the trapdoors $T_{W_1}, \ldots, T_{W_n}$.

Another possible solution is to generate a ciphertext and a trapdoor for the combination of keywords. This allows to generate only one trapdoor and so to reduce the number of tests needed.

To simplify the explanation, consider the case of two keywords. A document is characterized by the keywords W_1 and W_2. The sender generates the following ciphertexts

$$
\begin{aligned}
C_{W_1} =& \big[t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s \cdot g^{r_1}, h^{r_1}\big], \\
C_{W_2} =& \big[t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s \cdot g^{r_2}, h^{r_2}\big], \\
C_{W_1 \& W_2} =& \big[t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s \cdot g^{r_3}, h^{r_3}\big].
\end{aligned}
$$

So, if the receiver wants to search for both W_1 and W_2, it can create the trapdoor $T_{W_1 \& W_2} = \big[e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s, h^\rho), g^\rho, h^\rho\big]$.

Another solution is to encrypt the combination of W_1 and W_2 as

$$
\big[t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_1||W_2)^s \cdot g^{r_3}, h^{r_3})\big]
$$

and then create the trapdoor for $W_1||W_2$. However, in this way first of all the sender and the receiver should agree on an keyword ordering (e.g. lexicographic), so that it is used always the same concatenation (since using $W_1||W_2$ instead of $W_2||W_1$ would change the output of the hash). Then, the sender needs to compute also $H(\mathrm{pk}_S||\mathrm{pk}_R||W_1||W_2)^s$, while, for the previous solution, the sender has already computed $H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s$ and $H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s$, so it does not need to perform another hashing and exponentiation.

As final remark, let us note that, if we do not use the shared secret t in the encryption and in the trapdoor, then the server is able to generate valid ciphertexts corresponding to combinations of keywords. Indeed, given two ciphertexts $C_{W_1} = \big[H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s \cdot g^{r_1}, h^{r_1})\big]$ and $C_{W_2} = \big[H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s \cdot g^{r_2}, h^{r_2})\big]$, the server, just by multiplying them entry-by-entry, will obtain a valid ciphertext for the combination of the keywords. That is,

$$\begin{aligned} C_{W_1} * C_{W_2} &= \big[H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s \cdot g^{r_1}, h^{r_1})\big] * \big[H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s \cdot g^{r_2}, h^{r_2})\big] \\ &= \big[H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s \cdot g^{r_1+r_2}, h^{r_1+r_2})\big] = C_{W_1 \& W_2}. \end{aligned}$$

5 Security Proofs

In this section, we prove that our PAEKS scheme is CI-secure and (MU) fully TI-secure. Notice that, if we exchange the role of the `Encrypt` and `Trapdoor` algorithms, then we obtain a scheme TI-secure and (MU) fully CI-secure.

5.1 Trapdoor Indistinguishability

Before stating the security result, recall that if we assume that the DBDH problem over (G, G_T, e) is intractable, then also the CDH problem over G is intractable.

Theorem 1. *Under the DBDH assumption, our PAEKS scheme is (MU) fully TI-secure in a random oracle model.*

Proof. To prove the theorem, we show that, for the proposed scheme, winning the related game with a non-negligible advantage implies solving the DBDH problem with a non-negligible advantage. Assume that there is a PPT adversary $\mathcal{A}$ which breaks the trapdoor privacy of our scheme with a non-negligible advantage ϵ_T. We show in the following that there exists an algorithm $\mathcal{B}$ that is able to solve the DBDH problem with a non-negligible advantage.

Consider an instance of the DBDH problem, $(G, G_T, e, p, g, g^x, g^y, g^z, Z)$ where $x, y, z \in \mathbb{Z}_p$ are chosen randomly, and Z is either a random element of G_T or equal to $e(g, g)^{xyz}$. Let β be a bit such that $\beta = 0$ if $Z = e(g, g)^{xyz}$ and $\beta = 1$ otherwise. The goal of $\mathcal{B}$ is to guess the bit β and it does so by simulating the (MU) full TI-security game for $\mathcal{A}$ as follows.

Initialization $\mathcal{B}$ controls the random oracles that define the hash functions H and H_2 and sets the system parameter to $\mathrm{Param} = (G, G_T, e, p, g, H, H_2)$. Then, it select two random values $a, b \in \mathbb{Z}_p$ and set $\mathrm{pk}_R = g^a$, $\mathrm{pk}_S = g^b$, so $\mathrm{sk}_R = a$ and $\mathrm{sk}_S = b$. Therefore $h = g^{ab}$. Moreover, it selects another random value $u \in \mathbb{Z}_p$ and sets $t = g^u$. This implies that $H_2(g^{ab}) = u$. The last secret s is set to be equal to x, so $H_2(g^u) = x$. To simplify the notation, we set $v = ab$. Then, $\mathcal{B}$ calls $\mathcal{A}$ on input $(\mathrm{Param}, \mathrm{pk}_S, \mathrm{pk}_R)$.

Phase 1 In this phase, four oracles are involved. The oracle for H_2 only requires an element in G as input. The oracle for H requires in input a tuple in $G \times G \times \{0,1\}^*$. The oracles for encryption and trapdoor require in input a pair in $G \times \{0,1\}^*$. The number of queries for the different oracles is limited, specifically at most q_{H_2}, q_H, q_T, and q_C queries for oracles $\mathcal{O}_{H_2}$, $\mathcal{O}_H$, $\mathcal{O}_T$, and $\mathcal{O}_C$ respectively. To simplify the description of the game, we assume that the adversary would not issue a pair (g_i, W_i) to $\mathcal{O}_T$ with $g_i \neq g^a$ (or $\mathcal{O}_C$ with $g_i \neq g^b$) before issuing the following queries: $(g_i, \mathrm{pk}_R, W_i)$ (or $(\mathrm{pk}_S, g_i, W_i)$) to $\mathcal{O}_H$; g_i^a (or g_i^b) to $\mathcal{O}_{H_2}$; g^α to $\mathcal{O}_{H_2}$, with α the output of the previous query. To the hash oracles we associate two lists L_{H_2} and L_H (initially empty) collecting the outcomes of the hashes. The mentioned oracles operate as follows.

- Hash Oracle $\mathcal{O}_{H_2}$. Given an element $g_i \in G$ if there is an element in L_{H_2} of the form $\langle g_i, n_i \rangle$, $\mathcal{B}$ returns n_i. If $g_i = g^u$, then $\mathcal{B}$ aborts and outputs a random bit β' as its guess of β. If $g_i = g^{ab}$, then $\mathcal{B}$ sets $n_i = u$. Otherwise, it selects a random $n_i \in \mathbb{Z}_p$. The pair $\langle g_i, n_i \rangle$ is added to the list L_{H_2}. $\mathcal{B}$ returns $H_2(g_i) = n_i$ as the hash value of g_i to $\mathcal{A}$.

- Hash Oracle $\mathcal{O}_H$. Given a tuple $(\overline{\mathrm{pk}}_S, \overline{\mathrm{pk}}_R, W_i)$, if there is a tuple in L_H of the form $\langle (\overline{\mathrm{pk}}_S, \overline{\mathrm{pk}}_R, W_i), h_i, a_i, c_i \rangle$, then $\mathcal{B}$ returns h_i. Otherwise, $\mathcal{B}$ selects a random $a_i \in \mathbb{Z}_p$ and a biased $c_i \in \{0,1\}$ such that $\Pr[c_i = 0] = \delta$. Then $\mathcal{B}$ sets $h_i = g^z \cdot g^{a_i}$ if $c_i = 0$ and $h_i = g^{a_i}$ otherwise. The tuple $\langle (\overline{\mathrm{pk}}_S, \overline{\mathrm{pk}}_R, W_i), h_i, a_i, c_i \rangle$ is added to the list L_H. $\mathcal{B}$ returns $H(\overline{\mathrm{pk}}_S || \overline{\mathrm{pk}}_R || W_i) = h_i$ as the hash value of W_i to $\mathcal{A}$.

- Trapdoor Oracle $\mathcal{O}_T$. Given a pair $(\overline{\mathrm{pk}}_S, W_i)$, $\mathcal{B}$ retrieves from L_H the tuple $\langle (\overline{\mathrm{pk}}_S, \mathrm{pk}_R, W_i), h_i, a_i, c_i \rangle$. $\mathcal{B}$ selects a random $\rho_i \in \mathbb{Z}_p$. If $\overline{\mathrm{pk}}_S = g^a$, then $\mathcal{B}$ computes the trapdoor T_i as follows. If $c_i = 1$, then $\mathcal{B}$ sets $T_i = \left[e(g^u (g^x)^{a_i}, g^{v\rho_i}), g^{\rho_i}, g^{v\rho_i}\right]$. Otherwise, if $c_i = 0$, then $\mathcal{B}$ sets $T_i = \left[e(g^u (g^x)^{a_i}, g^{v\rho_i}) \cdot e(g^z, (g^x)^{v\rho_i}), g^{\rho_i}, g^{v\rho_i}\right]$. Notice that T_i is a correct well-distributed trapdoor since $H(\mathrm{pk}_S || \mathrm{pk}_R || W_i) = g^{a_i}$ in the first case and $H(\mathrm{pk}_S || \mathrm{pk}_R || W_i) = g^{a_i} \cdot g^z$ in the second one.

  Otherwise, if $\overline{\mathrm{pk}}_S \neq \mathrm{pk}_S$, then $\mathcal{B}$ sets $\overline{h} = (\overline{\mathrm{pk}}_S)^b$, and retrieves $\langle \overline{h}, n_i \rangle$ from L_{H_2}. It sets $\overline{t} = g^{n_i}$, then it retrieves $\langle \overline{t}, \overline{s} \rangle$ from L_{H_2}. Then it returns the trapdoor $T_i = \left[e(\overline{t} \cdot h_i^{\overline{s}}, \overline{h}^\rho), g^\rho, \overline{h}^\rho\right]$.

- Ciphertext Oracle $\mathcal{O}_C$. Given a pair $(\overline{\mathrm{pk}}_R, W_i)$, $\mathcal{B}$ retrieves the tuple $\langle (\mathrm{pk}_S, \overline{\mathrm{pk}}_R, W_i), h_i, a_i, c_i \rangle$ from L_H. If $c_i = 0$ then it aborts and outputs a random bit β' as its guess of β. Otherwise, it selects a random $r_i \in \mathbb{Z}_p$. If $\overline{\mathrm{pk}}_R = g^b$, then it returns the ciphertext $C_i = \left[C_{i,1}, C_{i,2}\right] = \left[g^{u+r_i} (g^x)^{a_i}, g^{vr_i}\right]$. Notice that C_i is a well-distributed ciphertext.

Otherwise, if $\overline{\text{pk}}_R \neq \text{pk}_R$, it sets $\overline{h} = (\overline{\text{pk}}_R)^a$, it retrieves $\langle \overline{h}, n_i \rangle$ from L_{H_2}. It sets $\overline{t} = g^{n_i}$, then it retrieves $\langle \overline{t}, \overline{s} \rangle$ from L_{H_2}. Then it returns the ciphertext $C_i = \big[C_{i,1}, C_{i,2}\big] = \big[\overline{t} \cdot h_i^{\overline{s}} \cdot g^r, \overline{h}^r\big]$.

Challenge The adversary $\mathcal{A}$ submits two keywords W_0^* and W_1^*, we assume that $(\text{pk}_S, \text{pk}_R, W_0^*)$ and $(\text{pk}_S, \text{pk}_R, W_1^*)$ have been queried to $\mathcal{O}_H$, but (pk_R, W_0^*) and (pk_R, W_1^*) have not been queried to $\mathcal{O}_C$. $\mathcal{B}$ retrieves from L_H the tuples $\langle (\text{pk}_S, \text{pk}_R, W_0^*), h_0^*, a_0^*, c_0^* \rangle$ and $\langle (\text{pk}_S, \text{pk}_R, W_1^*), h_1^*, a_1^*, c_1^* \rangle$. If $c_0^* = c_1^* = 1$, then it aborts and outputs a random bit β' as a guess of β. If $c_0^* = c_1^* = 0$, then let γ be a bit selected at random. Otherwise, let γ be the bit such that $c_\gamma^* = 0$. Notice that γ is uniformly distributed in $\{0, 1\}$. $\mathcal{B}$ then computes the trapdoor

$$T^* = \Big[\Big(Z \cdot e(g^x, g^y)^{a_\gamma^*} \cdot e(g, g^y)^u \Big)^v, g^y, (g^y)^v \Big] = \big[Q_1^*, Q_2^*, Q_3^*\big].$$

Notice that, if $Z = e(g, g)^{xyz}$ then

$$\begin{aligned} Q_1^* = & \Big(Z \cdot e(g^x, g^y)^{a_\gamma^*} \cdot e(g, g^y)^u \Big)^v = \Big(e(g^{xz}, g^y) \cdot e(g^{xa_\gamma^*}, g^y) \cdot e(g^u, g^y) \Big)^v \\ = & e(g^u \cdot g^{x(z + a_\gamma^*)}, g^y)^v = e(t \cdot H(\text{pk}_S || \text{pk}_R || W_\gamma^*)^s, h^y). \end{aligned}$$

Therefore T^* is a correct trapdoor. In this case, the random value chosen for the trapdoor corresponds to y. Otherwise the first entrance in Q_1^* is a random element of G_2. The tuple T^* is returned to the adversary.

Phase 2 $\mathcal{A}$ continues issuing queries to the oracles, with the restriction that it can not issue (pk_R, W_0^*) and (pk_R, W_1^*) to $\mathcal{O}_C$. Moreover, we can also assume that for the new queries to $\mathcal{O}_H$ we can always set the value c_i to be equal to 1.

Guess Finally, $\mathcal{A}$ outputs a bit γ'. If $\gamma' = \gamma$ then $\mathcal{B}$ outputs $\beta' = 0$, otherwise $\beta' = 1$.

We analyse now the success probability of $\mathcal{B}$. Denote by `abt` the event that $\mathcal{B}$ aborts during the game, this is divided into three events.

- $\texttt{abt}_0$: if $g_i = g^u$ in the simulation of $\mathcal{O}_{H_2}$. Since u was selected randomly over $\mathbb{Z}_p$, therefore determining g^u is either a random guess or, given that $H_2(g^{ab}) = u$, corresponds to solving the Computational Diffie-Hellman problem (CDH). Therefore, under some limitations on the number of queries q_{H_2}, we have that $\Pr[\texttt{abt}_0]$ is negligible.
- $\texttt{abt}_1$: if $c_i = 0$ in the simulation of $\mathcal{O}_C$. Each c_i is selected randomly and independently, therefore the probability that $\texttt{abt}_1$ does not happen is $\Pr[\overline{\texttt{abt}_1}] = (1 - \delta)^{q_C}$.

- $\mathtt{abt}_2$: if $c_0^* = c_1^* = 1$ in the generation of the challenge trapdoor. Therefore, $\Pr[\overline{\mathtt{abt}_2}] = 1 - (1-\delta)^2$.

So, the probability that $\mathcal{B}$ does not abort in the game is bounded by $\Pr[\overline{\mathtt{abt}}] = \Pr[\overline{\mathtt{abt}_0}] \cdot \Pr[\overline{\mathtt{abt}_1}] \cdot \Pr[\overline{\mathtt{abt}_2}] = \Pr[\overline{\mathtt{abt}_0}] \cdot (1-\delta)^{q_C}(1-(1-\delta)^2)$. With $\delta = 1 - \sqrt{\frac{q_C}{q_C+2}}$, the probability takes the maximal value $\Pr[\overline{\mathtt{abt}}] = \Pr[\overline{\mathtt{abt}_0}] \cdot \left(\frac{q_C}{q_C+2}\right)^{q_C/2} \cdot \frac{2}{q_C+2}$, approximately equal to $\Pr[\overline{\mathtt{abt}_0}] \cdot \frac{2}{q_C \cdot e}$ and thus non-negligible.

We have seen that, if $\beta = 0$ (i.e. $Z = e(g,g)^{xyz}$) and $\mathcal{B}$ does not abort, then the view of $\mathcal{A}$ is identically distributed as in a real attack. In this case, if $\mathcal{A}$ succeeds in breaking the trapdoor privacy of our scheme, then $\mathcal{B}$ succeeds in solving the DBDH problem instance. Note also that, if $\beta = 1$ then $\mathcal{A}$ acts on random inputs, so $\mathcal{B}$ effectively outputs a random guess, and thus the probability of guessing correctly is $\frac{1}{2}$. Therefore, the probability of guessing the bit β (and thus solving the DBDH problem) is:

$$
\begin{aligned}
\Pr[\beta' = \beta] =& \Pr[\beta' = \beta \mid \beta = 0]\Pr[\beta = 0] + \Pr[\beta' = \beta \mid \beta = 1]\Pr[\beta = 1] \\
=& \frac{1}{2}\left(\Pr[\beta' = \beta \mid \beta = 0] + \Pr[\beta' = \beta \mid \beta = 1]\right) \\
=& \frac{1}{2}\left(\Pr[\beta' = \beta \mid \beta = 0] + \frac{1}{2}\right) \\
=& \frac{1}{2}\left(\Pr[\beta' = \beta \mid \beta = 0 \wedge \mathtt{abt}]\Pr[\mathtt{abt}] + \Pr[\beta' = \beta \mid \beta = 0 \wedge \overline{\mathtt{abt}}]\Pr[\overline{\mathtt{abt}}] + \frac{1}{2}\right) \\
=& \frac{1}{2}\left(\frac{1}{2}(1 - \Pr[\overline{\mathtt{abt}}]) + (\epsilon_T + \frac{1}{2})\Pr[\overline{\mathtt{abt}}] + \frac{1}{2}\right) \\
=& \frac{1}{2}\epsilon_T \Pr[\overline{\mathtt{abt}}] + \frac{1}{2}.
\end{aligned}
$$

If ϵ_T is non-negligible, so it is $|\Pr[\beta' = \beta] - 1/2|$.

$\square$

5.2 Ciphertext indistinguishability

Recall that, if we asssume that the mDLIN problem over G is intractable, then also the CDH problem over G is intractable.

Theorem 2. *Under the mDLIN assumption, our PAEKS scheme has ciphertext indistinguishability under CKC and CKT attacks in random oracle model.*

Proof. To prove the theorem, we show that winning the related game with a non-negligible advantage implies solving the mDLIN problem with a non-negligible advantage. Assume that there is a PPT adversary $\mathcal{A}$ which breaks the ciphertext indistinguishability of our scheme with a non-negligible advantage ϵ_C. We want to show that we can build an algorithm $\mathcal{B}$ that is able to solve the mDLIN problem with a non-negligible advantage.

Consider an instance of the mDLIN problem $(G, G_T, e, p, g, g^x, g^y, g^{jx}, g^{k/y}, Z)$ where x, y, j, k are randomly chosen from $\mathbb{Z}_p$, and Z is either a random element of G or $Z = g^{j+k}$. Let β be a bit such that $\beta = 0$ if $Z = g^{j+k}$ and $\beta = 1$ otherwise. The goal of $\mathcal{B}$ is to guess the bit β and it does so by simulating the CI-security game for $\mathcal{A}$ as follows.

Initialization $\mathcal{B}$ controls the random oracles that define the hash functions H and H_2 and sets $\text{Param} = (G, G_T, e, p, g, H, H_2)$. Then it selects a random value $a \in \mathbb{Z}_p$ and it sets $\text{pk}_R = g^a$, $\text{pk}_S = g^{x/a}$, so $\text{sk}_R = a$ and $\text{sk}_S = x/a$. Therefore, the common secret h corresponds to g^x. Moreover, $\mathcal{B}$ selects another random value $u \in \mathbb{Z}_p$ and sets $t = g^u$. The last common secret is set as $s = y$. Therefore we have $H_2(g^x) = u$ and $H_2(g^u) = y$. Since we are in a single-user settings, we simplify the notation by writing $H(W)$ instead of $H(\text{pk}_S||\text{pk}_R||W)$. Then, $\mathcal{B}$ calls $\mathcal{A}$ on input $(\text{Param}, \text{pk}_S, \text{pk}_R)$.

Phase 1 $\mathcal{B}$ answers the adversary's queries with the same oracles and with the same assumptions considered in the TI case, but in a single-user setting. We describe the differences with the oracles of the previous game.

- The hash oracle $\mathcal{O}_{H_2}$ aborts if it is called on g^u.
- The action of the hash oracle $\mathcal{O}_H$ is as follows. Given a keyword W_i, it selects a random $a_i \in \mathbb{Z}_p$ and a biased $c_i \in \{0, 1\}$ such that $\Pr[c_i = 0] = \delta$. It sets $h_i = g^{k/y} \cdot g^{a_i}$ if $c_i = 0$, and $h_i = g^{a_i}$ otherwise. The tuple $\langle W_i, h_i, a_i, c_i \rangle$ is added to the list L_H (initially empty). It returns $H(W_i) = h_i$ as the hash value of W_i to $\mathcal{A}$.
- For the trapdoor oracle $\mathcal{O}_T$, given in input a keyword W_i, $\mathcal{B}$ retrieves the tuple $\langle W_i, h_i, a_i, c_i \rangle$ from L_H. If $c_i = 0$, it aborts and output a random bit β' as guess of β. In the other case (where $H(W_i) = g^{a_i}$) it selects a random ρ_i and outputs
$$T_i = \left[e(g^u (g^y)^{a_i}, (g^x)^{\rho_i}), g^{\rho_i}, (g^x)^{\rho_i}\right].$$
- Similarly, for the ciphertext oracle $\mathcal{O}_C$, given in input a keyword W_i, $\mathcal{B}$ retrieves the tuple $\langle W_i, h_i, a_i, c_i \rangle$ from L_H. If $c_i = 0$, it aborts and outputs a random bit $\beta\prime$ as guess of β. Otherwise, $\mathcal{B}$ selects a random r_i and outputs
$$C_i = \left[g^u (g^y)^{a_i} g^{r_i}, (g^x)^{r_i}\right].$$

Notice that both the ciphertext and trapdoor oracles' answers are legit.

Challenge When the adversary submits two keywords W_0^* and W_1^*, queried to $\mathcal{O}_H$ but not to $\mathcal{O}_T$ or $\mathcal{O}_C$, $\mathcal{B}$ retrieves the tuples $\langle W_0^*, h_0^*, a_0^*, c_0^* \rangle$ and $\langle W_1^*, h_1^*, a_1^*, c_1^* \rangle$ from L_H. If $c_0^* = c_1^* = 1$, then it aborts and outputs a random bit β' as a guess of β. If $c_0^* = 0$ or $c_1^* = 0$, it sets γ be the bit such that $c_\gamma^* = 0$, so $h_\gamma^* = g^{k/y} \cdot g^{a_\gamma^*}$.

If $c_0^* = c_1^* = 0$, then γ is selected at random. Notice that, as in the previous game, $\gamma = 0$ is uniformly distributed. $\mathcal{B}$ computes the ciphertext

$$C^* = \left[C_1^*, C_2^*\right] = \left[Z \cdot g^u (g^y)^{a_\gamma^*}, g^{xj}\right].$$

If $Z = g^{j+k}$ then $C_1^* = Z \cdot g^u (g^y)^{a_\gamma^*} = g^{j+k} g^u (g^y)^{a_\gamma^*} = t g^{y(a_\gamma^* + k/y)} g^j = tH(W_\gamma^*)^s g^j$, and $C_2^* = h^j$. In this case, the random element corresponds to j and C^* is a proper ciphertext. If Z is a random element of G_1, so it is C_1^*.

The tuple C^* is returned to the adversary.

Phase 2 $\mathcal{A}$ continues to issuing queries to the oracles, with the restriction that it cannot issue W_0^*, W_1^* to $\mathcal{O}_T$ nor $\mathcal{O}_C$. Moreover, we can also assume that for the new queries to $\mathcal{O}_H$ we can always set the value $c_i = 1$.

Guess Finally, $\mathcal{A}$ outputs a bit γ'. If $\gamma' = \gamma$ then $\mathcal{B}$ outputs $\beta' = 0$, otherwise $\beta' = 1$.

Denoted by $\texttt{abt}$ the event that $\mathcal{B}$ aborts during the game, the probability of this event is similar to the one in the previous game.

- $\texttt{abt}_0$: if $g_i = g^u$ in the simulation of $\mathcal{O}_{H_2}$. Since u was selected randomly over $\mathbb{Z}_p$, therefore determining g^u is either a random guess or, given that $H_2(g^x) = u$, corresponds to solving the Computational Diffie-Hellman problem (CDH), since the adversary knows $\mathrm{pk}_S = g^{x/a}$ and $\mathrm{pk}_R = g^a$. Therefore, under some limitations on the number of queries q_{H_2}, $\Pr[\texttt{abt}_0]$ is negligible.

- $\texttt{abt}_1$: if $c_i = 0$ in the simulation of $\mathcal{O}_C$ or $\mathcal{O}_T$. Each c_i is selected randomly and independently, therefore the probability that $\texttt{abt}_1$ does not happen is $\Pr[\overline{\texttt{abt}_1}] = (1-\delta)^{q_C + q_T}$.

- $\texttt{abt}_2$: if $c_0^* = c_1^* = 1$ in the generation of the challenge trapdoor. Therefore, $\Pr[\overline{\texttt{abt}_2}] = 1 - (1-\delta)^2$.

So, the probability that $\mathcal{B}$ does not abort in the game is bounded by

$$\Pr[\overline{\texttt{abt}}] = \Pr[\overline{\texttt{abt}_0}] \cdot \Pr[\overline{\texttt{abt}_1}] \cdot \Pr[\overline{\texttt{abt}_2}].$$

With $\delta = 1 - \sqrt{\frac{q_T + q_C}{q_T + q_C + 2}}$, we obtain

$$\Pr[\overline{\texttt{abt}}] = \Pr[\overline{\texttt{abt}_0}] \cdot \left(\frac{q_Q + q_C}{q_Q + q_C + 2}\right)^{(q_Q + q_C)/2} \cdot \frac{2}{q_Q + q_C + 2},$$

which is approximately equal to $\Pr[\overline{\texttt{abt}_0}] \cdot \frac{2}{(q_Q + q_C)e}$ and thus non-negligible. We have seen that, if $\beta = 0$ (i.e. $Z = g^{j+k}$) and $\mathcal{B}$ does not abort, then the view of $\mathcal{A}$ is identically distributed as in a real attack. In this case, if $\mathcal{A}$ succeeds in breaking the ciphertext privacy of our scheme, then $\mathcal{B}$ succeeds in solving the mDLIN problem instance. As before, if $\beta = 1$ then $\mathcal{A}$ acts on random inputs, so $\mathcal{B}$ effectively outputs a random guess, and thus the probability of guessing correctly is $\frac{1}{2}$. Therefore, the probability of guessing the bit β (and thus solving the mDLIN problem) is:

$$\begin{aligned}
\Pr[\beta' = \beta] =& \Pr[\beta' = \beta \mid \beta = 0]\Pr[\beta = 0] + \Pr[\beta' = \beta \mid b = 1]\Pr[\beta = 1] \\
=& \frac{1}{2}\left(\Pr[\beta' = \beta \mid \beta = 0] + \Pr[\beta' = \beta \mid \beta = 1]\right) \\
=& \frac{1}{2}\left(\Pr[\beta' = \beta \mid \beta = 0] + \frac{1}{2}\right) \\
=& \frac{1}{2}\left(\Pr[\beta' = \beta \mid \beta = 0 \wedge \mathtt{abt}]\Pr[\mathtt{abt}] + \Pr[\beta' = \beta \mid \beta = 0 \wedge \overline{\mathtt{abt}}]\Pr[\overline{\mathtt{abt}}] + \frac{1}{2}\right) \\
=& \frac{1}{2}\left(\frac{1}{2}(1 - \Pr[\overline{\mathtt{abt}}]) + (\epsilon_T + \frac{1}{2})\Pr[\overline{\mathtt{abt}}] + \frac{1}{2}\right) \\
=& \frac{1}{2}\epsilon_T \Pr[\overline{\mathtt{abt}}] + \frac{1}{2}.
\end{aligned}$$

If ϵ_C is non-negligible, so it is $|\Pr[\beta' = \beta] - 1/2|$. $\square$

6 Conclusions

In this work we presented a new PAEKS scheme, which not only randomized the ciphertext but also the trapdoor. We proved that our scheme is fully TI-secure and CI-secure (or fully CI-secure and TI-secure if we swap the encryption and trapdoor algorithms). A further work is to study whether this scheme is also fully CI-secure, thanks to the randomzation introduced in both ciphertexts and trapdoors.

References

[1] Bethencourt, J., Sahai, A., and Waters, B.: Ciphertext-policy attribute-based encryption. In 2007 IEEE symposium on security and privacy (SP'07) (pp. 321-334). IEEE (2007).

[2] Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G.: Public key encryption with keyword search. In International conference on the theory and applications of cryptographic techniques (pp. 506-522). Springer, Berlin, Heidelberg (2004).

[3] Boneh, D., and Franklin, M.: Identity-based encryption from the Weil pairing. In Annual international cryptology conference (pp. 213-229). Springer, Berlin, Heidelberg (2001).

[4] Boneh, D., Lynn, B., and Shacham, H.: Short signatures from the Weil pairing. In International conference on the theory and application of cryptology and information security (pp. 514-532). Springer, Berlin, Heidelberg (2001).

[5] Byun, J.W., Rhee, H.S., Park, H., Lee, D.H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data, in: SDM 2006, LNCS 4165, 2006, pp. 75–83 (2006).

[6] Demertzis, I., Chamani, J.G., Papadopoulos, D., Papamanthou, C.: Dynamic searchable encryption with small client storage. In: 27th Annual Network and Distributed System Security Symposium. NDSS 2020, San Diego, California, USA, 23–26 February 2020. The Internet Society (2020).

[7] He, D., Ma, M., Zeadally, S., Kumar, N., Liang, K.: Certificateless public key authenticated encryption with keyword search for industrial internet of things. IEEE Trans. Ind. Inform. 14(8), 3618–3627 (2018).

[8] Huang, Q., Li, H.: An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf. Sci. 403, 1–14 (2017).

[9] Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Comput. Commun. 32(2), 394–396 (2009).

[10] Joux, A.: A one round protocol for tripartite Diffie–Hellman. In International algorithmic number theory symposium (pp. 385-393). Springer, Berlin, Heidelberg (2000).

[11] Lu, Y., Wang, G., Li, J.: Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement. Inf. Sci. 479, 270–276 (2019).

[12] Mascia, C., Sala, M. and Villa, I.: A survey on functional encryption. Advances in Mathematics of Communications (2021).

[13] Noroozi, M., Eslami, Z.: Public key authenticated encryption with keyword search: revisited. IET Inf. Secur. 13(4), 336–342 (2019).

[14] Noroozi, M., Karoubi, I., Eslami, Z.: Designing a secure designated server identity-based encryption with keyword search scheme: still unsolved. Ann. des Telecommunications 73(11–12), 769–776 (2018).

[15] Qin, B., Chen, Y., Huang, Q., Liu, X., Zheng, D.: Public-key authenticated encryption with keyword search revisited: security model and constructions. Inf. Sci. 516, 515–528 (2020).

[16] Qin, B., Cui, H., Zheng, X., and Zheng, D.: Improved security model for public-key authenticated encryption with keyword search. In International Conference on Provable Security (pp. 19-38). Springer, Cham (2021).

[17] Soleimanian, A., Khazaei, S.: Publicly verifiable searchable symmetric encryption based on efficient cryptographic components. Des. Codes Cryptography 87(1), 123– 147 (2019)

[18] Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society (2000).

[19] Yau, W.-C., Heng, S.-H., Goi, B.-M.: Off-Line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) ATC 2008. LNCS, vol. 5060, pp. 100–105. Springer, Heidelberg (2008).

[20] Yau,W.,Phan,R.C.,Heng,S.,Goi,B.: Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. Int. J. Comput. Math. 90(12), 2581–2587 (2013).