Research Article

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

Searchable encryption with randomized ciphertext and randomized keyword search

https://doi.org/10.1515/jmc-2023-0029
received September 05, 2023; accepted October 18, 2023

Abstract: The notion of public-key encryption with keyword search (PEKS) was introduced to search over encrypted data without performing any decryption. In this article, 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 *a priori*. Our scheme is *Ciphertext-Indistinguishabiltity* secure in the single-user setting and *Trapdoor-Indistinguishability* secure in the multi-user setting with a stronger security, i.e., with *multi-trapdoor*.

Keywords: searchable encryption, PAEKS scheme, cloud cryptography, bilinear pairings

MSC 2020: 94A60 Cryptography, 14H52 Elliptic curves, 94A62, authentication and secret sharing, 68W40 analysis of algorithms

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 data owners lose 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 being uploaded to the 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 few years, many prominent cryptographic primitives have been proposed for achieving secure and efficient cloud data usage, such as searchable encryption (SE) [1,2]. SE 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 from the plaintexts (the documents) and generating searchable ciphertexts corresponding to these keywords [3–7]. Then, data receivers can search uploaded encrypted documents to find those containing a keyword by generating a trapdoor to send to the server. Once the trapdoor has been received, the server runs an algorithm to test which documents contain the searched keyword. If there is a match, then the server returns the associated encrypted document.

*** Corresponding author: Riccardo Longo,** Department of Mathematics, University of Trento, via Sommarive 14, 38123, Povo (Trento), Italy; Center for Cybersecurity, Fondazione Bruno Kessler, Via Sommarive 18, 38123, Povo (Trento), Italy, e-mail: rlongo@fbk.eu
Marco Calderini: Department of Mathematics, University of Trento, via Sommarive 14, 38123, Povo (Trento), Italy, e-mail: marco.calderini@unitn.it
Massimiliano Sala: Department of Mathematics, University of Trento, via Sommarive 14, 38123, Povo (Trento), Italy, e-mail: maxsalacodes@gmail.com
Irene Villa: Department of Mathematics, University of Trento, via Sommarive 14, 38123, Povo (Trento), Italy; Department of Mathematics, Excellence Department 2023-2027, University of Genova, via Dodecaneso 35, 16146, Genova, Italy, e-mail: irene1villa@gmail.com
ORCID: Marco Calderini 0000-0002-6817-3421; Riccardo Longo 0000-0002-8739-3091; Massimiliano Sala 0000-0002-7266-5146; Irene Villa 0000-0001-6381-4712

The first SE schemes that appeared in the literature use a symmetric setting [2]. In 2004, Boneh et al. [1] proposed the first public-key encryption with keyword search (PEKS). In a multi-user setting, PEKS [1] allows any user to encrypt keywords for searching, by designated searching key holders.

However, PEKS schemes are vulnerable to offline keyword guessing attacks (KGAs) [8,9]. That is, given a trapdoor, the adversary can generate a ciphertext of a guessing keyword and then test whether it matches the trapdoor. If the keyword space has low entropy, this attack is very efficient. Indeed, several PEKS schemes are shown to be insecure against KGAs [5,8–12].

Public-key authenticated encryption with keyword search (PAEKS) was proposed in 2017 by Huang and Li [13] to defend against KGAs. Its security model guarantees two security goals: cipher-keyword indistinguishability (CI-security) and trapdoor indistinguishability (TI-security). The first refers to the fact that an attacker is not able to distinguish which keyword is associated to a ciphertext, even when this keyword is one (randomly chosen by the challenger) of two alternatives controlled by the attacker itself. Conversely, trapdoor indistinguishability states that an attacker is not able to distinguish which keyword is associated to a trapdoor, even when this keyword is one (randomly chosen by the challenger) of two alternatives controlled by the attacker itself. In the *standard* setting, the attacker may ask queries to a trapdoor oracle and a ciphertext oracle as long as they are not related to the challenge, while in the Full Security setting, this restriction is relaxed.

Recently, Noroozi and Eslami [14] showed that the PAEKS scheme in the study by Huang and Li [13] is not secure in the multi-user setting, and Qin et al. [6] showed that it is not secure in the multi-cipher-keyword setting. Both proposed some adjustments to their scheme. Qin et al. [15] proposed a PAEKS scheme, and they proved that their scheme is secure in the multi-cipher-keyword setting for CI-security and in the multi-user setting. However, in the study by Qin et al. [15] 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 addition, Emura [16], Emura introduced a PAEKS scheme, where each keyword is converted into an extended keyword, and a PEKS scheme is then used for extended keywords. The author defined trapdoor privacy by formalizing indistinguishability against keyword guessing attack (IND-IKGA) and proved that the proposed scheme is secure under this notion. As remarked by Emura [16], IND-IKGA does not imply full TI-security.

In this article, we propose a PAEKS scheme with an improved TI-security model with respect to the study by Qin et al. [15]. 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, i.e., encrypting two times the same keyword (or creating the trapdoor for the same keyword) will produce different results, in contrast with the study by Qin et al. [15] where the trapdoor is not randomized. Obviously, if the cloud server receives two trapdoors, both testing positive on the same ciphertext, then the server learns that the two corresponding queries are the same, even if the trapdoors look different. Yet, in our system, the cloud server is not able to recognize *a priori* that two encrypted trapdoors correspond to the same query. This is a significant advancement w.r.t. previous schemes.

Note also that, regarding our discussion on security, if we exchange the role of the encryption and trapdoor algorithm, we would obtain a PAEKS scheme that is CI-secure in the multi-user setting and TI-secure in the single-user setting.

This article is structured as follows: in Section 2, we provide some preliminary notions that are useful to understand the rest of the article; 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 and G_T be two multiplicative cyclic groups of prime order p. An admissible pairing is defined as a map $e : G \times G \to G_T$ that satisfies 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 [17], attribute-based encryption schemes [18], key-agreement protocols [19], signature schemes [20]. Many schemes based on pairings can be found in a recent survey on functional encryption in the study by Mascia et al. [21].

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 2.1. (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.

Note that, when we are considering an admissible bilinear map, solving the decisional version of the above problem is easy, so we have to adapt it as follows.

Definition 2.2. (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, and 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 the CDH is also intractable.

Definition 2.3. (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 $g^x, g^y, g^{xr}, g^{ys} \in G$, for x, y, s, and r randomly chosen from $\mathbb{Z}_p$, distinguish g^{r+s} from a random element of G.

As in the study by Huang and Li [13], we consider the modified decisional linear (mDLIN) problem, which is defined as below.

Definition 2.4. (mDLIN) Given a generator $g \in G$ and the elements $g^x, g^y, g^{jx}, g^{k/y} \in G$, for x, y, j, and k randomly chosen from $\mathbb{Z}_p$, distinguish g^{j+k} 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 the CDH is also intractable. Indeed, suppose that given g^a and g^b, we are able to compute efficiently g^{ab}. Thus, from g^y and $g^{k/y}$, we are able to recover g^k. Then, we can distinguish $Z = g^{j+k}$ from a random element by computing Z/g^k and then solving the CDH problem between Z/g^k and g^x and check if it coincides with g^{jx}.

3 Preliminaries on public-key SE schemes

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

3.1 PEKS

A PEKS consists of the following (probabilistic) polynomial-time algorithms [1].
- `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 [1].

This type of scheme is vulnerable to the inside KGA. Indeed, to recover the keyword contained in a trapdoor T_W, an honest-but-curious cloud server could check whether W' equals 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 carry out the KGA in a reasonably short time.

To address this issue, the notion of PAEKS was introduced in 2017 [13].

3.2 PAEKS

A 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.
- `KeyGen`$_S$(Param): given the system parameter, the sender's key pair generation algorithm outputs a pair of public and secret keys (pk_S, sk_S) for the sender.
- `KeyGen`$_R$(Param): given the system parameter, the receiver's key pair generation algorithm outputs a pair of public and secret keys (pk_R, sk_R) for the receiver.
- `Encrypt`(W, sk_S, 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.
- `Trapdoor`(W, pk_S, 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.
- `Test`(pk_S, pk_R, $C_{W'}$, T_W): given the sender'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 by Huang and Li [13], 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 to PEKS, security of PAEKS requires that there is no probabilistic polynomial-time adversary that could distinguish trapdoors or ciphertexts. Therefore, a semantic security model for PAEKS includes both CI-security and TI-security or Trapdoor Privacy. Related to the security of PAEKS schemes, we recall the notation presented in the study by Qin et al. [15], adapted 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 setting, an adversary may have the following two abilities to attack a PAEKS scheme.

3.3.1 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 $C_W = \mathrm{Encrypt}(W, \mathrm{sk}_S, \overline{\mathrm{pk}}_R)$. Formally, CKC attacks are modelled by giving the adversary $\mathcal{A}$ access to a ciphertext oracle $\mathrm{Encrypt}_{\mathrm{sk}_S}(\cdot,\cdot)$, 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 $C_W = \mathrm{Encrypt}(W, \mathrm{sk}_S, \overline{\mathrm{pk}}_R)$.

3.3.2 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 $T_W = \mathrm{Trapdoor}(W, \overline{\mathrm{pk}}_S, \mathrm{sk}_R)$. Similar to the previous case, CKT attacks are modelled by giving the adversary $\mathcal{A}$ access to a trapdoor oracle $\mathrm{Trapdoor}_{\mathrm{sk}_R}(\cdot,\cdot)$, 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 $T_W = \mathrm{Trapdoor}(W, \overline{\mathrm{pk}}_S, \mathrm{sk}_R)$.

Clearly, the adversary's access to the aforementioned oracles has to be restricted in some trivial instances. Let W_0^* and W_1^* be the challenge keywords, then, in the CI-security model, the adversary cannot request the trapdoors of the challenge keywords for the target public keys, lest the challenge become trivial. In many PAEKS schemes, such as [13], 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^*. Qin et al. [15], in their study, considered *full CKC attacks*, where this limitation is removed.

Similarly, for TI-security, the adversary cannot request the ciphertexts of the challenge keywords. In addition, in this case, many PAEKS schemes (see [13,15]) impose other limitations such as the adversary is not allowed to request trapdoors corresponding to either W_0^* or W_1^*. In this article, 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. Note that in a multiuser setting, the adversary can also choose a public key to give as extra input to the oracles, while in the single-user setting, this public key is fixed.

3.3.3 (MU) full TI-security model

We describe the *full TI-security game* for an adversary $\mathcal{A}$ in the multi-user setting.

(1) Initialization: Given a security parameter λ, the challenger runs the setup algorithm to generate the global system parameter Param. Then, it runs $\text{KeyGen}_S(\text{Param})$ and $\text{KeyGen}_R(\text{Param})$ to generate the target sender's key pair $(\text{pk}_S, \text{sk}_S)$ and the target receiver's key pair $(\text{pk}_R, \text{sk}_R)$, respectively. The challenger invokes the adversary $\mathcal{A}$ on input $(\text{Param}, \text{pk}_S, \text{pk}_R)$.
(2) Phase 1: The adversary is allowed to adaptively issue queries to the following oracles for polynomially many times.
   – **Trapdoor oracle** O_T: Given a keyword W and a public key $\overline{\text{pk}}_S$, the oracle computes the corresponding trapdoor T_W with respect to $\overline{\text{pk}}_S$ and sk_R, and returns T_W to $\mathcal{A}$.
   – **Ciphertext oracle** O_C: Given a keyword W and a public key $\overline{\text{pk}}_R$, the oracle computes the corresponding ciphertext C_W with respect to sk_S and $\overline{\text{pk}}_R$, 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 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 $T_{W_\beta^*} \leftarrow \text{Trapdoor}(W_\beta^*, \text{pk}_S, \text{sk}_R)$, and returns $T_{W_\beta^*}$ to $\mathcal{A}$.
(4) Phase 2: The adversary continues to issue queries to O_T and 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 in correctly distinguishing the scheme's trapdoors as $\text{Adv}^T_{\mathcal{A}}(\lambda) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 3.1. ([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 $\text{Adv}^T_{\mathcal{A}}(\lambda)$ is negligible in λ.

3.3.4 Standard CI-security model

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

(1) Initialization: Given a security parameter λ, the challenger generates Param and prepares pk_S and pk_R as in the previous Game. It then invokes the adversary $\mathcal{A}$ on input $(\text{Param}, \text{pk}_S, \text{pk}_R)$.
(2) Phase 1: The adversary issues queries to oracles O_T and O_C as before, but in the single-user setting (no public key is given in input to the oracles since it is implicitly set to pk_S and pk_R, respectively).
(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 $C_{W_\beta^*} \leftarrow \text{Encrypt}(W_\beta^*, \text{sk}_S, \text{pk}_R)$, and returns $C_{W_\beta^*}$ to $\mathcal{A}$.
(4) Phase 2: The adversary continues to issue queries to O_T and 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 in correctly distinguishing the ciphertexts as $\text{Adv}^C_{\mathcal{A}}(\lambda) = |\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 3.2. (Standard 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 $\text{Adv}^C_{\mathcal{A}}(\lambda)$ is negligible in λ.

4 The new scheme

In this section, we describe a new public-key authenticated SE scheme. For the sake of brevity, we condense the two algorithms KeyGen_S and 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 the 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 $\text{Param} = (G, G_T, e, p, g, H, H_2)$.
- KeyGen. Given Param, the algorithm produces the following pair of keys for sender and receiver: $(\text{pk}_S, \text{sk}_S) = (g^a, a)$ and $(\text{pk}_R, \text{sk}_R) = (g^b, b)$, 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, and s are obtained. The sender selects a random $r \in \mathbb{Z}_p^*$ and outputs the pair:
$$C_W = [C_1, C_2] = [t \cdot H(\text{pk}_S||\text{pk}_R||W)^s \cdot g^r, h^r].$$
- `Trapdoor`. Given a keyword $W \in \{0,1\}^*$, pk_S, and sk_R, the common secrets h, t, and s are obtained. The receiver selects a random $\rho \in \mathbb{Z}_p$ and outputs the tuple:
$$T_W = [Q_1, Q_2, Q_3] = [e(t \cdot H(\text{pk}_S||\text{pk}_R||W)^s, h^\rho), g^\rho, h^\rho].$$
- `Test`. Given in input any trapdoor $T_{W'} = [Q_1, Q_2, Q_3]$, corresponding to a keyword W', and any ciphertext $C_W = [C_1, C_2]$, 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

$$e(C_1, Q_3) = e(t \cdot H(\text{pk}_S||\text{pk}_R||W)^s \cdot g^r, h^\rho) = e(t \cdot H(\text{pk}_S||\text{pk}_R||W)^s, h^\rho) \cdot e(g^r, h^\rho),$$

and

$$Q_1 \cdot e(C_2, Q_2) = e(t \cdot H(\text{pk}_S||\text{pk}_R||W')^s, h^\rho) e(h^r, g^\rho) = e(t \cdot H(\text{pk}_S||\text{pk}_R||W')^s, h^\rho) e(g^r, h^\rho),$$

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

Remark 1. The aim of the `KeyGen` algorithm is to generate three secrets only known by the sender and the receiver. Note 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 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.

For example, consider the case of two keywords. A document is characterized by the keywords W_1 and W_2. The sender generates the following ciphertexts:

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

So, if the receiver wants to search for both W_1 and W_2, a solution would be to create the trapdoor:

$$T_{W_1\&W_2} = [e(t \cdot H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s, h^\rho), g^\rho, h^\rho].$$

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

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

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 a keyword ordering (e.g. lexicographic), so that it is always used with 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 a 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:

$$C_{W_1} = [H(\mathrm{pk}_S||\mathrm{pk}_R||W_1)^s \cdot g^{r_1}, h^{r_1}], \quad C_{W_2} = [H(\mathrm{pk}_S||\mathrm{pk}_R||W_2)^s \cdot g^{r_2}, h^{r_2}],$$

the server, just by multiplying them entry-by-entry, will obtain a valid ciphertext for the combination of the keywords. That is,

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

5 Security proofs

In this section, we prove that our PAEKS scheme is CI-secure and (MU) fully TI-secure. Note 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 the CDH problem over G is also intractable (see Definition 2.2).

Theorem 5.1. *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 probabilistic polynomial time adversary $\mathcal{A}$ that 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.

5.1.1 Initialization

$\mathcal{B}$ controls the random oracles that define the hash functions H and H_2 and sets the system parameters to Param = $(G, G_T, e, p, g, H, H_2)$. Then, it selects two random values $a, b \in \mathbb{Z}_p$ and sets $\mathrm{pk}_R = g^a$, $\mathrm{pk}_S = g^b$, 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 implicitly set to be equal to x, so that $H_2(g^u) = x$. To simplify the notation, we set $v = ab$. Then, $\mathcal{B}$ calls $\mathcal{A}$ on input (Param, pk_S, and pk_R).

5.1.2 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 to at most q_{H_2}, q_H, q_T, and q_C queries for the oracles O_{H_2}, O_H, O_T, and 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 O_T with $g_i \neq g^a$ (or 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 O_H; g_i^a (or g_i^b) to O_{H_2}; and g^a to O_{H_2}, where a is the output of the previous query. To the hash oracles, we associate two lists L_{H_2} and L_H (initially empty) collecting the outputs of the hashes. The mentioned oracles operate as follows:

- Hash Oracle 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 O_H. Given a tuple $(\overline{\mathrm{pk}}_S, \overline{\mathrm{pk}}_R, W_i)$, then $\mathcal{B}$ returns h_i if there is a tuple in L_H of the form $\langle (\overline{\mathrm{pk}}_S, \overline{\mathrm{pk}}_R, W_i), h_i, a_i, c_i \rangle$. 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. Finally, $\mathcal{B}$ returns to $\mathcal{A}H(\overline{\mathrm{pk}}_S \| \overline{\mathrm{pk}}_R \| W_i) = h_i$ as the hash value of W_i.
- Trapdoor oracle O_T. Given a pair $(\overline{\mathrm{pk}}_S, W_i)$, $\mathcal{B}$ selects a random $\rho_i \in \mathbb{Z}_p$ and retrieves from L_H the tuple $\langle (\overline{\mathrm{pk}}_S, \mathrm{pk}_R, W_i), h_i, a_i, c_i \rangle$. 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 $T_i = [e(g^u (g^x)^{a_i}, g^{v\rho_i}), g^{\rho_i}, g^{v\rho_i}]$.
  - If $c_i = 0$, then $\mathcal{B}$ sets $T_i = [e(g^u (g^x)^{a_i}, g^{v\rho_i}) \cdot e(g^z, (g^x)^{v\rho_i}), g^{\rho_i}, g^{v\rho_i}]$.

Note that T_i is a well-distributed trapdoor: in fact, in the first case, we have $H(\mathrm{pk}_S \| \mathrm{pk}_R \| W_i) = g^{a_i}$, and in the second case, $H(\mathrm{pk}_S \| \mathrm{pk}_R \| W_i) = g^{a_i} \cdot g^z$.

- 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 $\bar{t} = g^{n_i}$, then it retrieves $\langle \bar{t}, \bar{s} \rangle$ from L_{H_2}. Then, it returns the trapdoor $T_i = [e(\bar{t} \cdot h_i^{\bar{s}}, \overline{h}^{\rho}), g^{\rho}, \overline{h}^{\rho}]$.
- Ciphertext oracle O_C. Given a pair $(\overline{\mathrm{pk}}_R, W_i)$, $\mathcal{B}$ retrieves from L_H the tuple $\langle (\mathrm{pk}_S, \overline{\mathrm{pk}}_R, W_i), h_i, a_i, c_i \rangle$ and selects a random $r_i \in \mathbb{Z}_p$. We distinguish two cases:
  - If $\overline{\mathrm{pk}}_R \neq \mathrm{pk}_R$, $\mathcal{B}$ sets $\overline{h} = (\overline{\mathrm{pk}}_R)^a$ and retrieves $\langle \overline{h}, n_i \rangle$ from L_{H_2}. Then, it sets $\bar{t} = g^{n_i}$ and retrieves $\langle \bar{t}, \bar{s} \rangle$ from L_{H_2}. Finally, $\mathcal{B}$ returns the ciphertext $C_i = [C_{i,1}, C_{i,2}] = [\bar{t} \cdot h_i^{\bar{s}} \cdot g^{r_i}, \overline{h}^{r_i}]$.
  - If $\overline{\mathrm{pk}}_R = g^b$, then $\mathcal{B}$ checks the value of c_i retrieved from L_H. If $c_i = 0$, then it aborts and outputs a random bit β' as its guess of β. Otherwise, it returns the ciphertext $C_i = [C_{i,1}, C_{i,2}] = [g^{u+r_i} (g^x)^{a_i}, g^{vr_i}]$. Note that C_i is a well-distributed ciphertext.

5.1.3 Challenge

The adversary $\mathcal{A}$ submits two keywords W_0^* and W_1^*, and we assume that $(\mathrm{pk}_S, \mathrm{pk}_R, W_0^*)$ and $(\mathrm{pk}_S, \mathrm{pk}_R, W_1^*)$ have been queried to O_H, but (pk_R, W_0^*) and (pk_R, W_1^*) have not been queried to O_C. $\mathcal{B}$ retrieves from L_H the

tuples $\langle(\mathrm{pk}_S, \mathrm{pk}_R, W_0^*), h_0^*, a_0^*, c_0^*\rangle$ and $\langle(\mathrm{pk}_S, \mathrm{pk}_R, W_1^*), h_1^*, a_1^*, c_1^*\rangle$. 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 $c_\gamma^* = 0$. Note that γ is uniformly distributed in $\{0, 1\}$. $\mathcal{B}$ then computes the trapdoor

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

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

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

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.

5.1.4 Phase 2

$\mathcal{A}$ continues issuing queries to the oracles, with the restriction that it cannot issue (pk_R, W_0^*) and (pk_R, W_1^*) to O_C. In this phase, $\mathcal{B}$ sets $c_i = 1$ for all new queries to O_H.

5.1.5 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, which is divided into three events.

- abt_0: if $g_i = g^u$ in the simulation of 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 CDH. Therefore, under some limitations on the number of queries q_{H_2}, we have that $\Pr[\mathrm{abt}_0]$ is negligible.
- abt_1: if $c_i = 0$ and $\overline{\mathrm{pk}}_R = g^b$ in the simulation of O_C. We can overestimate this probability by assuming that $\overline{\mathrm{pk}}_R = g^b$ in every call to O_C. Each c_i is selected randomly and independently, therefore a lower bound to the probability that abt_1 does not happen is $\Pr[\overline{\mathrm{abt}_1}] \leq (1 - \delta)^{q_C}$.
- abt_2: if $c_0^* = c_1^* = 1$ in the generation of the challenge trapdoor. Therefore, $\Pr[\overline{\mathrm{abt}_2}] = 1 - (1 - \delta)^2$.

So, the probability that $\mathcal{B}$ does not abort in the game can be computed as follows:

$$\Pr[\overline{\mathrm{abt}}] = \Pr[\overline{\mathrm{abt}_0}] \cdot \Pr[\overline{\mathrm{abt}_1}] \cdot \Pr[\overline{\mathrm{abt}_2}] \leq \Pr[\overline{\mathrm{abt}_0}] \cdot (1 - \delta)^{q_C}(1 - (1 - \delta)^2).$$

If we set $\delta = 1 - \sqrt{\frac{q_C}{q_C + 2}}$, then we have that the probability takes the maximal value

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

which is approximately $\Pr[\overline{\mathrm{abt}_0}] \cdot \frac{2}{q_C \cdot e}$ 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:

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

Thus, if the advantage ε_T of the adversary $\mathcal{A}$ is non-negligible, then the advantage $|\Pr[\beta' = \beta] - 1/2|$ of $\mathcal{B}$ is also non-negligible.

5.2 Ciphertext indistinguishability

Recall that, if we assume that the mDLIN problem over G is intractable, then the CDH problem over G is also intractable (see Definition 2.4).

Theorem 5.2. *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 probabilistic polynomial time 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, and 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. □

5.2.1 Initialization

$\mathcal{B}$ controls the random oracles that define the hash functions H and H_2 and sets up the parameters as $\mathrm{Param} = (G, G_T, e, p, g, H, H_2)$. Then, $\mathcal{B}$ selects a random value $a \in \mathbb{Z}_p$, and it sets $\mathrm{pk}_R = g^a$ and $\mathrm{pk}_S = g^{x/a}$, so $\mathrm{sk}_R = a$ and $\mathrm{sk}_S = x/a$. 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 setting, we simplify the notation by writing $H(W)$ instead of $H(\mathrm{pk}_S||\mathrm{pk}_R||W)$. Then, $\mathcal{B}$ calls $\mathcal{A}$ on input $(\mathrm{Param}, \mathrm{pk}_S, \mathrm{pk}_R)$.

5.2.2 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 O_{H_2} aborts if it is called on g^u.

– The action of the hash oracle 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 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 outputs a random bit β' as a guess of β. In the other case (where $H(W_i) = g^{a_i}$), it selects a random ρ_i and outputs

$$T_i = [e(g^u(g^y)^{a_i}, (g^x)^{\rho_i}), g^{\rho_i}, (g^x)^{\rho_i}].$$

– Similarly, for the ciphertext oracle 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 β' as a guess of β. Otherwise, $\mathcal{B}$ selects a random r_i and outputs

$$C_i = [g^u(g^y)^{a_i} g^{r_i}, (g^x)^{r_i}].$$

Note that both the ciphertext and trapdoor oracles' answers are correctly distributed.

5.2.3 Challenge

When the adversary submits two keywords W_0^* and W_1^*, queried to O_H but not to O_T or O_C, $\mathcal{B}$ retrieves from L_H the tuples $\langle W_0^*, h_0^*, a_0^*, c_0^*\rangle$ and $\langle W_1^*, h_1^*, a_1^*, c_1^*\rangle$. If $c_0^* = c_1^* = 1$, then it aborts and outputs a random bit β' as a guess of β. If $c_0^* = 0$ 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. Note that, as in the previous game, γ is uniformly distributed. $\mathcal{B}$ computes the ciphertext

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

If $Z = g^{j+k}$, then we have $C_1^* = g^{j+k} g^u (g^y)^{a_\gamma^*} = g^u g^{y(a_\gamma^* + k/y)} g^j = tH(W_\gamma^*)^s g^j$, 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.

5.2.4 Phase 2

$\mathcal{A}$ continues to issue queries to the oracles, with the restriction that it cannot issue W_0^* and W_1^* to O_T nor O_C. As in the previous game, in this phase, $\mathcal{B}$ sets $c_i = 1$ for all new queries to O_H.

5.2.5 Guess

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

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

- abt_0: if $g_i = g^u$ in the simulation of 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 CDH since the adversary knows $\text{pk}_S = g^{x/a}$ and $\text{pk}_R = g^a$. Therefore, under some limitations on the number of queries q_{H_2}, $\Pr[\text{abt}_0]$ is negligible.
- abt_1: if $c_i = 0$ in the simulation of O_C or O_T. Each c_i is selected randomly and independently, therefore the probability that abt_1 does not happen is $\Pr[\overline{\text{abt}_1}] = (1 - \delta)^{q_C + q_T}$.
- abt_2: if $c_0^* = c_1^* = 1$ in the generation of the challenge trapdoor. Therefore, $\Pr[\overline{\text{abt}_2}] = 1 - (1 - \delta)^2$.

Table 1: Comparison of our scheme with the one proposed in in the study by Qin et al. [15]

	Ciphertext	Trapdoor	TI-security	CI-security
[15]	Randomized	Deterministic	Standard in multi-user	Fully in multi-user
This article	Randomized	Randomized	Fully in multi-user	Standard in single-user

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

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

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

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

which is approximately equal to $\Pr[\overline{\text{abt}_0}]\cdot \frac{2}{(q_Q+q_C)e}$ 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, just like in the previous game:

$$\Pr[\beta' = \beta] = \frac{1}{2}\varepsilon_T \Pr[\overline{\text{abt}}] + \frac{1}{2}.$$

Thus, if the advantage ε_C of the adversary $\mathcal{A}$ is non-negligible, then the advantage $|\Pr[\beta' = \beta] - 1/2|$ of $\mathcal{B}$ is also non-negligible.

To better understand the improvements of the proposed scheme, we present a comparison with the PAEKS scheme presented in the study by Qin et al. [15], see Table 1.

6 Conclusions and open problems

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). We also discussed how to use our SE scheme for combination of keywords.

Future work could be towards two directions:

- To modify our scheme in order to obtain trapdoors that implement disjunctive queries, i.e., that can be satisfied simultaneously by subsets of a set of keywords. This would reduce the amount of information leaked to the cloud server when it performs the test since it is possible to tune the search more finely and disclose only the overall match between trapdoors and ciphertexts.
- To extend our scheme to provide also CI-security in the multi-user setting.

Acknowledgements: This work has been accepted for presentation at CIFRIS23, the Congress of the Italian association of cryptography “De Componendis Cifris.” The first, second, and last authors are members of the INdAM Research Group GNSAGA.

Funding information: This work has been partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. Funding by the MUR Excellence Department Project awarded to Dipartimento di Matematica, UniversitÃă di Genova, CUP

D33C23001110001, and by the European Union within the program NextGenerationEU. The third author acknowledges support from Ripple's University Blockchain Research Initiative.

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and approved its submission.

Conflict of interest: Prof. Massimiliano Sala is the Editor-in-Chief of the Journal of Mathematical Cryptology and was not involved in the review process of this article.

Data availability statement: Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

References

[1] Boneh D, Crescenzo GD, Ostrovsky R, and Persiano G. Public key encryption with keyword search. In: International conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer; 2004. p. 506–22.
[2] Song DX, Wagner DA, Perrig A. Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy. Berkeley, CA, USA: IEEE Computer Society; 2000. p. 44–55.
[3] Demertzis I, Chamani JG, 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.
[4] 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. 2018;14(8):3618–27.
[5] Noroozi M, Karoubi I, Eslami Z. Designing a secure designated server identity-based encryption with keyword search scheme: still unsolved. Ann des Telecommun. 2018;73(11-12):769–76.
[6] Qin B, Chen Y, Huang Q, Liu X, Zheng D. Public-key authenticated encryption with keyword search revisited: security model and constructions. Inf Sci. 2020;516: 515–28.
[7] Soleimanian A, Khazaei S. Publicly verifiable searchable symmetric encryption based on efficient cryptographic components. Des Codes Cryptography 2019;87(1):123–47.
[8] Byun JW, Rhee HS, Park H, Lee DH. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: SDM 2006, LNCS 4165, 2006; 2006. p. 75–83.
[9] Jeong IR, Kwon JO, Hong D, Lee DH. Constructing PEKS schemes secure against keyword guessing attacks is possible? Comput Commun. 2009;32(2):394–6.
[10] 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. 2019;479:270–6.
[11] 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 MG, Sandnes FE, Yang LT, Ma J, (eds.) ATC 2008. LNCS. Vol. 5060. Heidelberg: Springer; 2008. p. 100–5.
[12] Yau W, Phan RC, Heng S, Goi B Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. Int J Comput Math. 2013;90(12):2581–7.
[13] Huang Q, Li H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf Sci. 2017;403:1–14.
[14] Noroozi M, Eslami Z. Public key authenticated encryption with keyword search: revisited. IET Inf Secur. 2019;13(4):336–42.
[15] Qin B, Cui H, Zheng X, Zheng D. Improved security model for public-key authenticated encryption with keyword search. In: International Conference on Provable Security. Cham: Springer; 2021. p. 19–38.
[16] Emura K. Generic construction of public-key authenticated encryption with keyword search revisited: stronger security and efficient construction. Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop; 2022.
[17] Boneh D, Franklin M. Identity-based encryption from the Weil pairing. In: Annual international cryptology conference. Berlin, Heidelberg: Springer; 2001. p. 213–29.
[18] Bethencourt J, Sahai A, and Waters B. Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP'07). IEEE; 2007. p. 321–34.
[19] Joux A. A one round protocol for tripartite Diffie–Hellman. In: International algorithmic number theory symposium. Berlin, Heidelberg: Springer; 2000. p. 385–93.
[20] Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. In: International Conference on the theory and Application of Cryptology and Information Security. Berlin, Heidelberg: Springer; 2001. p. 514–32.
[21] Mascia C, Sala M, Villa I. A survey on functional encryption. Adv Math Commun. 2023;17(5):1251–89.