IET Information Security

Research Article

Public key authenticated encryption with keyword search: revisited

ISSN 1751-8709
Received on 21st June 2018
Accepted on 27th September 2018
E-First on 28th February 2019
doi: 10.1049/iet-ifs.2018.5315
www.ietdl.org

Mahnaz Noroozi[1], *Ziba Eslami*[2] ✉

[1]*Department of Computer and Data Science, Shahid Beheshti University, G.C., Tehran, Iran*
[2]*Cyberspace Research Institute, Shahid Beheshti University, G.C., Tehran, Iran*
✉ *E-mail: z_eslami@sbu.ac.ir*

Abstract: In 2017, the notion of *public key authenticated encryption with keyword search* (PAEKS) and its security model was defined by Huang and Li. Their main motivation was providing security against inside *keyword guessing attacks* (KGA). They also proposed a concrete PAEKS scheme secure in their proposed model. In this study, the authors first show that their security model has an important drawback and therefore, cannot handle multi-user settings. As such settings are a necessity in the public-key environment, it is vital to improving the model to capture multiple users. This is what they do in the first part of this study. Then, they consider Huang and Li's PAEKS scheme and prove that it is not secure against inside (and even outside) KGA. Finally, they propose a modified scheme to fix the problem without any additional communication or computation costs. They further prove that the new scheme is secure in the improved model.

1 Introduction

On the plus side, the advent of cloud computing technology has enabled users to remotely maintain and manage their data at considerably lower costs. The downside, however, is that owners now lose their full control over data and consequently are forced to take additional steps to protect the confidentiality of their sensitive information. The straightforward approach to solve this problem is to apply encryption before uploading the data to the cloud server. Unfortunately, this simple technique gives rise to some concerns since we are now faced with a new challenge: how can data owners efficiently perform search operations on encrypted data stored on the cloud? The question has turned out to be a hot research topic in recent years and has attracted the attention of several researchers.

To answer this question, in 2000 [1], Song *et al.* introduced a cryptographic notion called *searchable encryption* (SE), which allows keyword search over encrypted data. There are three entities involved in an SE scheme:

i. Data senders (also called writers) who generate a ciphertext C^i and cipher-keywords $\{C^i_{w^i_j}\}$ corresponding to each document D^i and its keywords $\{w^i_j\}$, respectively, and then upload them on the cloud server(s). We assume that K_C is the key used for ciphertext/cipher-keyword generation. Encapsulation of documents $\{D^i\}$ into $\{C^i\}$ can be done using any secure encryption scheme. However, the focus of attention in SE is the algorithm used for encryption of keywords which should be devised in such a way that efficient searching becomes possible.
ii. Data receivers (also called readers) who can search uploaded (encrypted) documents $\{C^i\}$ to find the ones containing some keyword w. To do so, they use some secret key K_T to generate a trapdoor T_w and send it to the cloud server.
iii. The server(s) who manages the encrypted database. Upon receiving the trapdoor T_w (corresponding to some keyword w which is unknown to the server), the server runs a test algorithm against each cipher-keyword $C^i_{w^i_j}$ to test if it contains the same keyword as T_w or not. If there is a match between the keyword encapsulated in T_w and the one in $C^i_{w^i_j}$, then the server returns the associated ciphertext C^i (corresponding to the document D^i which is unknown to the server).

The first SE schemes that appeared in the literature use a symmetric setting where the same key is used in keyword encryption as well as trapdoor generation. In other words, there is a shared secret key $K(= K_T = K_C)$ between each corresponding data sender and receiver. This property makes *searchable symmetric encryption* (SSE) suitable only for outsourcing scenarios. The interested reader can find more on SSE schemes in [2–5].

To incorporate SE in public key setting, Boneh *et al.* introduced in 2004, the notion of *public key encryption with keyword search* (PEKS). Here, searchable ciphertexts are generated using the public key of intended recipients while trapdoor generation uses the corresponding secret keys. Therefore, PEKS schemes are also applicable to data sharing scenarios. Fig. 1 depicts the general framework of SE in both symmetric and public key versions. However, in this study, we are only concerned with asymmetric settings.

The generation and security of ciphertexts $\{C^i\}$ have already been extensively considered in the context of encryption schemes and is usually left out in SE. In a PEKS, related security definitions should further capture indistinguishability of cipher-keywords $\{C^i_{w^i_j}\}$. In other words, we expect that they leak as little information about their associated keywords as possible to a polynomially-bounded adversary. Nevertheless, a well-known fact about PEKS schemes is their vulnerability to *keyword guessing attack* (KGA). This attack was first launched by Byun *et al.* [6] and exploits the fact that in real-world applications, the keyword space is usually small. In a KGA, an attacker who obtains a trapdoor can guess the searched keyword and verify the accuracy of this guess in an off-line manner by encrypting that keyword (using the corresponding public keys) and using the test algorithm. Therefore, at least a cloud server (or other inside entities in the cloud management services who can observe trapdoors) might launch KGA. As a consequence, the literature enhanced the security requirements to encompass indistinguishability of trapdoors against KGA [7].

There are a few research studies in the literature aim to withstand the keyword guessing attack [8]. Some of the researchers [7, 9, 10] try to suppress the attack only against outside adversaries, i.e. anyone other than the server itself (who still can perform the attack). In [9], the server possesses its own public/

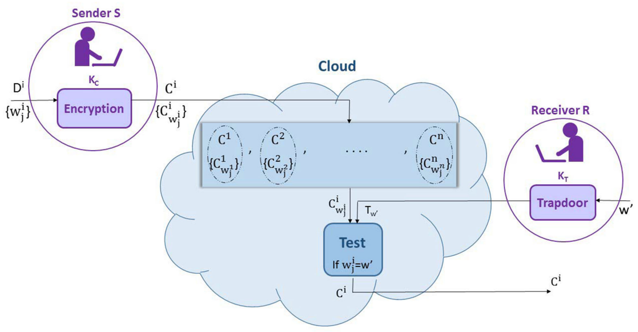

Fig. 1 *General framework of SE; in SSE: $K_C = K_T = K$; in PEKS: $K_C = Pk_R$ (public key of R), $K_T = Sk_R$ (secret key of R)*

secret keys and these keys get involved in the related algorithms in such a way that the protocol resists outside KGA. The authors of [7, 10] improved [9] in order to enhance security and efficiency. Some other papers considered both inside and outside keyword guessing attacks (KGA). In [11], the authors proposed a solution by forcing data senders to register their keywords to data receivers before encrypting them. However, due to the necessity of data receiver to be online on the one hand and the required communications for keyword registrations, on the other hand, this approach is impractical. In [12], another solution is proposed in which no one except the data receiver can verify the correspondence between cipher-keywords and trapdoors. However, this approach has no efficiency in terms of communication and computational costs. In 2013, Xu *et al.* [13] proposed a public key encryption with a fuzzy keyword search scheme in which the severity of the problem is reduced by making sure that each trapdoor corresponds to more than one keyword.

In another attempt to make PEKS secure against inside KGA, in 2017, Huang and Li [14] defined the notion of *public key authenticated encryption with keyword search* (PAEKS) and presented a security model for it. In PAEKS, the secret key of data senders is used in encrypting keywords (see Fig. 1 with K_C= {secret key of S, public key of R} and K_T= {public key of S, secret key of R}). Therefore, Huang and Li [14] claimed that in PAEKS, the servers or other adversaries can no longer encrypt possible keywords and consequently launch KGA. They also presented a concrete PAEKS scheme secure in their proposed model.

1.1 Our contributions

Our study mainly concerns the results of Huang and Li [14] (denoted hereafter for short by HL-PAEKS). Here is a list of our contributions in detail:

i. We show that the proposed security model in HL-PAEKS has a vital deficiency and as a consequence, it cannot handle multi-user settings. To be applicable in settings with multiple users, we noticed that the security model should assume that there are many users of the scheme besides the two attacked users, i.e. the sender P_S and the receiver P_R. Therefore, P_S can use appropriate public keys to encrypt and authenticate keywords not only for P_R but also for any other possible data receiver. Similarly, P_R can generate trapdoors using the public key of P_S or any other data sender. Accordingly, proper oracle accesses should be given to adversaries in the security model to capture these capabilities. Indeed, the problem with the proposed security model of [14] is that such oracle accesses are not considered.
ii. We define a new security model for PAEKS to capture multi-user settings.
iii. We show that the proposed PAEKS scheme in [14] cannot achieve its main purpose, i.e. it is not secure against inside (and even outside) KGA.
iv. We improve the scheme of Huang and Li to overcome these problems and prove its security in the new model. Our improvement is achieved without any additional communication or computation costs.

1.2 Organisation of the paper

The rest of this paper is organised as follows. Section 2 briefly covers the preliminaries of the paper including the concept of PAEKS. In Section 3, we explain the flaw of the proposed security model in HL-PAEKS and propose a new security model for PAEKS. Then, in Section 4, we show that the proposed concrete construction in HL-PAEKS is not secure against inside/outside KGA (contrary to its main purpose). We propose a modified version of the scheme of Huang and Li in Section 5 which overcomes the problem. In Section 6, we prove the security of our scheme in the new model and analyse its performance. Finally, Section 7 concludes the paper.

2 Preliminaries

In this section, we cover the preliminaries needed in the study. First, the definition of bilinear maps and some complexity assumptions are given. Then, we review the notion of PAEKS introduced in [14].

2.1 Bilinear maps

Let G_1 and G_T be two cyclic groups of prime order p. A map $e: G_1 \times G_1 \to G_T$ is called a bilinear map if it satisfies the following properties:

i. *Bilinearity*: $e(g^x, h^y) = e(g, h)^{xy}$ for all $g, h \in G_1$ and $x, y \in \mathbb{Z}$.
ii. *Non-degeneracy*: There exist $g, h \in G_1$ such that $e(g, h) \neq 1$ where 1 is the identity of G_T.
iii. *Computability*: There exists an efficient algorithm to compute $e(g, h)$ for any $g, h \in G_1$.

2.2 Complexity assumptions

Throughout this section, G_1 and G_T are two cyclic groups of prime order p, g is a generator of G_1 and $e: G_1 \times G_1 \to G_T$ is a bilinear map.

Definition 1: (decisional bilinear Diffie–Hellman (DBDH) problem): Given $(G_1, G_T, e, p, g, g^x, g^y, g^z, Z)$, where x, y, z are randomly chosen from $\mathbb{Z}_p$, determine whether Z is equal to $e(g, g)^{xyz} \in G_T$ or is a random element of G_T.

Definition 2: (DBDH assumption): For any polynomial-time algorithm $\mathscr{A}$, there exists a negligible function negl such that $| \Pr [\mathscr{A}(G_1, G_T, e, p, g, g^x, g^y, g^z, e(g, g)^{xyz}) = 1] -$
$\Pr [\mathscr{A}(G_1, G_T, e, p, g, g^x, g^y, g^z, e(g, g)^r) = 1] | \leq \mathrm{negl}(\lambda)$.

Definition 3: (modified decision linear (mDLIN) problem): Given $(G_1, G_T, e, p, g, g^x, g^y, g^{ry}, g^{s/x}, Z)$, where x, y, r, s are randomly chosen from $\mathbb{Z}_p^*$, determine whether Z equals g^{r+s} or is a random element.

Definition 4: (mDLIN assumption [14]*):* For any polynomial-time algorithm $\mathscr{A}$, there exists a negligible function negl such that

$$\begin{aligned} &| \Pr [\mathscr{A}(G_1, G_T, e, p, g, g^x, g^y, g^{ry}, g^{s/x}, g^{r+s}) = 1] \\ &\quad - \Pr [\mathscr{A}(G_1, G_T, e, p, g, g^x, g^y, g^{ry}, g^{s/x}, g^z) = 1] | \leq \mathrm{negl}(\lambda) \end{aligned}$$

2.3 Public key authenticated encryption with keyword search (PAEKS)

Recall from the introduction that KGA is an important issue in PEKS schemes. In [14], the authors introduce the notion of PAEKS in order to make PEKS withstand inside KGA. Their idea is to incorporate data sender's secret key in creating cipher-keywords so

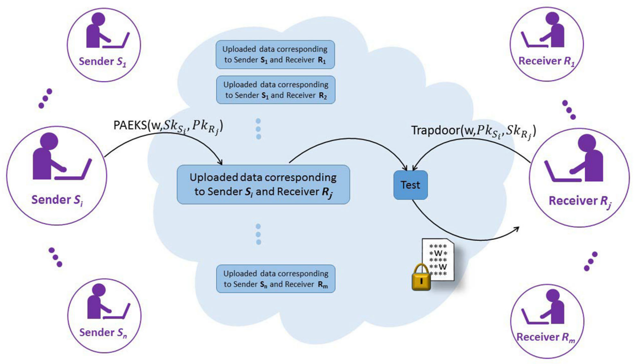
Sender S_1
Sender S_i
Sender S_n
Receiver R_1
Receiver R_j
Receiver R_m
Uploaded data corresponding to Sender S_1 and Receiver R_1
Uploaded data corresponding to Sender S_1 and Receiver R_2
Uploaded data corresponding to Sender S_i and Receiver R_j
Uploaded data corresponding to Sender S_n and Receiver R_m
PAEKS(w,Sk_{S_i}, Pk_{R_j})
Trapdoor(w,Pk_{S_i},Sk_{R_j})
Test

Fig. 2 *Schematic representation of encrypted cloud data in PAEKS*

that servers (who do not have the appropriate secret key) can no longer launch KGA.

In this section, we provide a presentation of the concept of PAEKS as defined in [14] which consists of the following algorithms:

- *Setup*$(\lambda) \to$ *Param*. Performed in a trusted way so that everyone in the system would trust the generated parameters, with input the security parameter λ, this algorithm outputs the global parameters of the system *Param*.
- $KeyGen_S(Param) \to (Pk_S, Sk_S)$. Performed by a data sender with input Param, this algorithm generates his public/secret key pair (Pk_S, Sk_S).
- $KeyGen_R(Param) \to (Pk_R, Sk_R)$. Performed by a data receiver with input *Param*, this algorithm generates his public/secret key pair (Pk_R, Sk_R).
- $PAEKS(w, Sk_S, Pk_R) \to C_w$. Performed by a data sender with input his secret key Sk_S, a data receiver's public key Pk_R, and a keyword w, this algorithm outputs a cipher-keyword C_w.
- $Trapdoor(w, Pk_S, Sk_R) \to T_w$. Performed by a data receiver with input his secret key Sk_R, a data sender's public key Pk_S, and a keyword w, this algorithm outputs a trapdoor T_w.
- $Test(T_w, C_{w'}, Pk_S, Pk_R) \to 1 \text{ or } 0$. Performed by the server with input a trapdoor T_w, a cipher-keyword $C_{w'}$, a data sender's public key Pk_S, and a data receiver's public key Pk_R, this algorithm returns 1 if $C_{w'}$ and T_w correspond the same keyword and 0 otherwise.

Fig. 2 provides a schematic representation of encrypted cloud data in PAEKS.

3 Security model of PAEKS

We noticed that the PAEKS security model proposed in [14] does not consider multiple users. In this section, we first explain the security requirements of a PAEKS scheme and describe where the deficiency in the security model of HL-PAEKS stems from. Then, we proceed to define a new security model for PAEKS schemes which captures multi-user settings as well.

3.1 Necessary security requirements in PAEKS

In PAEKS schemes, with regard to the security of cipher-keywords, we need to certify that PAEKS (w, Sk_S, Pk_R) does not reveal any information about w unless its corresponding trapdoor is available [15]. More precisely, distinguishing an encryption of the keyword w_0 from an encryption of the keyword w_1 should be infeasible for every active adversary who is able to obtain trapdoors and cipher-keywords corresponding to keywords of his choice (except for w_0 and w_1). Note that encryption of keywords in PAEKS, the same as generating trapdoors, needs some secret key and cannot be publicly performed. Therefore, the adversary should be equipped with appropriate oracle access in the security model to simulate such disclosure of information in real-world.

As explained in the Introduction, another concern in the literature is indistinguishability of trapdoors against KGA [7]. It means that, after obtaining a trapdoor, an adversary should not be able to link the trapdoor to any keywords, even under the assumption that the keyword space is small. Again, distinguishing T_{w_0} from T_{w_1} should be infeasible for every active adversary who is able to obtain trapdoors and cipher-keywords corresponding to keywords of his choice (except for w_0 and w_1).

Note that usually two types of adversaries are considered: insider and outsider. Inside adversary refers to the server while an outside adversary is any other entity (except for the data sender and receiver).

3.2 Problem in the security model of HL-PAEKS

All the above considerations have been fully considered in the HL-PAEKS model. What is ignored in the security model of [14] is the fact that in a multi-user setting, it should be assumed that there are other users of the scheme besides the attacked users P_S and P_R. This is because, in practice, any adversary may access some information corresponding to any other user P_i and use this information in attacking the intended ones (P_S and P_R). Besides, these other users (P_i s) themselves can be a potential adversary for P_S and P_R and hence, their information should not affect the security of the scheme. Therefore, the model should consider accessing this information for the adversary in order to be applicable in the real world. It is noteworthy that similar considerations have been taken into account in the context of signcryption [16].

Therefore, in multi-user searchable public-key encryption, P_S can encrypt keywords using any data receiver's public key. This means that the adversary should be equipped with the power to choose the data receiver when accessing cipher-keyword generation oracle of P_S as well. Therefore, we must define $\mathcal{O}_{\text{PAEKS}}(w, Pk)$ (instead of $\mathcal{O}_{\text{PAEKS}}(w)$ in HL-PAEKS model). Similarly, P_R can generate trapdoors using any data sender's public key. Therefore, we must define $\mathcal{O}_{\text{Trapdoor}}(w, Pk)$ (instead of $\mathcal{O}_{\text{Trapdoor}}(w)$ in the HL-PAEKS model).

3.3 Improved security model for PAEKS

We now formally define the improved security games played between a challenger and an adversary while the differences with the HL-PAEKS model are marked in bold. An implicit assumption made here (as in [14]) is that the model applies to 'static' adversaries, i.e. the attacked public keys are fixed at the beginning of the game.

3.3.1 Game 1: cipher-keyword indistinguishability: Let λ be the security parameter and let $\mathcal{A}$ be an adversary.

- *Initialisation*. The challenger runs the Setup(λ) algorithm to get the global parameters of the system Param and then, runs $\text{KeyGen}_S(\text{Param})$ and $\text{KeyGen}_R(\text{Param})$ to generate the data sender and the data receiver's key pairs (Pk_S, Sk_S) and (Pk_R, Sk_R), respectively. $(\text{Param}, Pk_S, Pk_R)$ are given to the adversary $\mathcal{A}$.
- *Phase 1*. In this phase, $\mathcal{A}$ can access the following oracles for a polynomially-bounded number of iterations. Such queries can be made adaptively so that the answers to the previous queries might affect subsequent ones.

  ○ **$\mathcal{O}_{\text{PAEKS}}(w, Pk)$. Given a keyword w and a public key Pk, this oracle outputs PAEKS (w, Sk_S, Pk).**

  ○ **$\mathcal{O}_{\text{Trapdoor}}(w, Pk)$. Given a keyword w and a public key Pk, this oracle outputs Trapdoor (w, Pk, Sk_R).**

  Note that in [14], we have $\mathcal{O}_{\text{PAEKS}}(w)$ and $\mathcal{O}_{\text{Trapdoor}}(w)$ instead.)
- *Challenge*. When phase 1 ends, $\mathcal{A}$ outputs two keywords (w_0^*, w_1^*) with the restriction that $\mathcal{O}_{\text{PAEKS}}(w_i^*, Pk_R)$ and $\mathcal{O}_{\text{Trapdoor}}(w_i^*, Pk_S)$ for $i = 0, 1$ have never been queried in phase 1.

Now, the challenger chooses a random bit $b \in \{0,1\}$ and computes $C_{w_b^*} = \mathrm{PAEKS}(w_b^*, Sk_S, Pk_R)$ and gives it to $\mathscr{A}$.

- *Phase 2.* In this phase, the adversary can continue to access the oracles in phase 1 with the restriction that $\mathcal{O}_{\mathrm{PAEKS}}(w_i^*, Pk_R)$ and $\mathcal{O}_{\mathrm{Trapdoor}}(w_i^*, Pk_S)$ should never be queried for $i = 0, 1$.
- *Response.* When phase 2 ends, $\mathscr{A}$ returns a bit $b' \in \{0,1\}$ as his guess about b and wins the game if $b = b'$.

We define $\mathrm{Adv}_{\mathscr{A}}^{C}(\lambda) = \left| \Pr[b = b'] - (1/2) \right|$ as the advantage of $\mathscr{A}$ in distinguishing the cipher-keywords of PAEKS.

3.3.2 Game 2: trapdoor indistinguishability: Let λ be the security parameter and $\mathscr{A}$ be an adversary who performs the keyword guessing attack.

- *Initialisation.* The challenger runs the Setup(λ) algorithm to get the global parameters of the system *Param* and then, runs KeyGen$_S$(*Param*) and KeyGen$_R$(*Param*) to generate the data sender and the data receiver's key pairs (Pk_S, Sk_S) and (Pk_R, Sk_R), respectively. $(\mathrm{Param}, Pk_S, Pk_R)$ are given to the adversary $\mathscr{A}$.
- *Phase 1.* The same as phase 1 of Game 1, $\mathscr{A}$ can adaptively access the oracles $\boldsymbol{\mathcal{O}_{\mathrm{PAEKS}}(w, Pk)}$ **and** $\boldsymbol{\mathcal{O}_{\mathrm{Trapdoor}}(w, Pk)}$ for a polynomially-bounded number of iterations.
- *Challenge.* When phase 1 ends, $\mathscr{A}$ outputs two keywords (w_0^*, w_1^*) with the restriction that $\mathcal{O}_{\mathrm{PAEKS}}(w_i^*, Pk_R)$ and $\mathcal{O}_{\mathrm{Trapdoor}}(w_i^*, Pk_S)$ for $i = 0, 1$ have never been queried in phase 1. Now, the challenger chooses a random bit $b \in \{0,1\}$ and computes $T_{w_b^*} = \mathrm{Trapdoor}(w_b^*, Pk_S, Sk_R)$ and gives it to $\mathscr{A}$.
- *Phase 2.* In this phase, the adversary can continue to access the oracles in phase 1 with the restriction that $\mathcal{O}_{\mathrm{PAEKS}}(w_i^*, Pk_R)$ and $\mathcal{O}_{\mathrm{Trapdoor}}(w_i^*, Pk_S)$ should never be queried for $i = 0, 1$.
- *Response.* When phase 2 ends, $\mathscr{A}$ returns a bit $b' \in \{0,1\}$ as his guess about b and wins the game if $b = b'$.

We define $\mathrm{Adv}_{\mathscr{A}}^{T}(\lambda) = \left| \Pr[b = b'] - (1/2) \right|$ as the advantage of $\mathscr{A}$ in distinguishing the trapdoors of PAEKS.

Definition 5: A PAEKS scheme is semantically secure against adaptive chosen keyword attacks if the following holds:

- It satisfies cipher-keyword indistinguishability which states that for any polynomial-time attacker $\mathscr{A}$, $\mathrm{Adv}_{\mathscr{A}}^{C}(\lambda)$ is a negligible function.
- It satisfies trapdoor indistinguishability against KGA which states that for any polynomial-time attacker $\mathscr{A}$, $\mathrm{Adv}_{\mathscr{A}}^{T}(\lambda)$ is a negligible function.

4 Insecurity of Huang and Li's scheme against KGA

Recall that the authors of [14] introduced the notion of PAEKS in order to overcome the problem of insecurity against inside KGA. However, their concrete PAEKS scheme is unfortunately insecure against KGA for both insiders and outsiders. In the following, after reviewing their construction, we prove this claim.

The HL-PAEKS scheme [14] consists of the following algorithms:

Setup(λ): Based on the security parameter λ, two cyclic groups G_1 and G_T of prime order p along with a random generator g of G_1 are chosen. Then, a bilinear pairing $e: G_1 \times G_1 \to G_T$ and a cryptographic hash function $H: \{0,1\}^* \to G_1$ are selected. $\mathrm{Param} = \{G_1, G_T, p, g, e, H\}$ is returned.

KeyGen$_S$(Param): $x \in_R \mathbb{Z}_p$ is chosen as the secret key of the sender (i.e. Sk_S) and $Pk_S = g^x$ is computed as his public key. (Pk_S, Sk_S) is returned by this algorithm.

KeyGen$_R$(Param): $y \in_R \mathbb{Z}_p$ is chosen as the secret key of the receiver (i.e. Sk_R) and $Pk_R = g^y$ is computed as his public key. (Pk_R, Sk_R) is returned by this algorithm.

PAEKS(w, Sk$_S$, Pk$_R$): $r \in_R \mathbb{Z}_p$ is selected and $c_w^1 = H(w)^{Sk_S} \cdot g^r$, $c_w^2 = Pk_R^r$ are computed. The cipher-keyword $C_w = (c_w^1, c_w^2)$ is returned.

Trapdoor(w, Pk$_S$, Sk$_R$): The trapdoor is output as $T_w = e(H(w)^{Sk_R}, Pk_S)$.

Test$(T_w, C_{w'} = (c_{w'}^1, c_{w'}^2), Pk_S, Pk_R)$: 1 is returned if $e(c_{w'}^1, Pk_R) = T_w \cdot e(g, c_{w'}^2)$ and 0 otherwise.

Theorem 1: The HL-PAEKS scheme does not provide security against inside/outside KGA.

Proof: As described in Section 3, security against KGA for a PAEKS scheme ensures that the advantage of any polynomial-time attacker $\mathscr{A}$ in winning game 2 (i.e. $\mathrm{Adv}_{\mathscr{A}}^{T}(\lambda)$) is a negligible function. Therefore, we show that in HL-PAEKS, there exists an adversary who can win game 2 with a non-negligible advantage.

Let S, R and $\mathscr{A}$ be a data sender, a data receiver, and an adversary, respectively. $\mathscr{A}$ plays game 2 with the challenger as follows:

- *Initialisation:* The challenger runs the Setup(λ) algorithm to get the global parameters of the system *Param* and then, runs KeyGen$_S$(*Param*) and KeyGen$_R$(*Param*) to generate the data sender and the data receiver's key pairs (Pk_S, Sk_S) and (Pk_R, Sk_R), respectively. Then, he gives $(\mathrm{Param}, Pk_S, Pk_R)$ to $\mathscr{A}$.
- *Phase 1:* $\mathscr{A}$ requests $\mathcal{O}_{\mathrm{PAEKS}}(w_0, Pk_{\mathscr{A}})$ for an arbitrary chosen w_0 and receives $C_{w_0} = (c_{w_0}^1, c_{w_0}^2)$. Then, he computes $\mathrm{Temp} := (c_{w_0}^1 / (c_{w_0}^2)^{Sk_{\mathscr{A}}^{-1}})$ and $T = e(\mathrm{Temp}, Pk_R)$.
- *Challenge:* $\mathscr{A}$ outputs two keywords (w_0, w_1), where w_1 is another arbitrarily chosen keyword. Then, the challenger chooses a random bit $b \in \{0,1\}$ and computes $T_{w_b} = \mathrm{Trapdoor}(w_b, Pk_S, Sk_R)$ and gives it to $\mathscr{A}$.
- *Response:* $\mathscr{A}$ checks whether T_{w_b} is equal to T or not. If yes, he returns $b' = 0$ and otherwise, $b' = 1$.

We claim that $b' = b$ and $\mathscr{A}$ wins the game with $\mathrm{Adv}_{\mathscr{A}}^{T}(\lambda) = (1/2)$. This is because

$c_{w_0}^1 = H(w_0)^{Sk_S} \cdot g^r$, $\quad c_{w_0}^2 = Pk_{\mathscr{A}}^r = g^{r \cdot Sk_{\mathscr{A}}}$ (for a random r) $\Rightarrow \mathrm{Temp} = H(w_0)^{Sk_S} \Rightarrow T = e(H(w_0)^{Sk_S}, Pk_R) = e(H(w_0), g)^{Sk_S \cdot Sk_R} = e(H(w_0)^{Sk_R}, Pk_S)$.

Therefore, T is indeed the correct trapdoor corresponding to w_0, i.e. $T = T_{w_0}$. Note that since $\mathscr{A}$ does not take advantage of anything that only a server can do, this adversary can be either an insider or an outside one. This completes the proof. □

5 Improved PAEKS scheme

In this section, we modify the scheme of Huang and Li [14] in order to overcome its drawback and make it secure in a multi-user setting against KGA. The improved PAEKS scheme consists of the following (probabilistic) polynomial-time algorithms: (the differences with HL-PAEKS scheme are highlighted)

- *Setup:* Performed in a trusted way so that everyone in the system would trust the generated parameters.

  ○ *Input:* The security parameter λ.
  ○ *Process:*

  i. Chooses two cyclic groups G_1 and G_T of prime order p.

ii. **Chooses two random generators g and h of G_1.**
iii. Chooses a bilinear pairing $e: G_1 \times G_1 \to G_T$.
iv. Chooses a cryptographic hash function $H: \{0,1\}^* \to G_1$.

o *Output:* The global parameters of the system Params $= \{G_1, G_T, p, g, h, e, H\}$.

- *KeyGen$_S$:* Performed by the data sender.

o *Input: Params.*
o *Process:*

i. **Chooses $x \in_R \mathbb{Z}_p$ as the sender's secret key Sk_S and computes his public key $Pk_S = h^x$.**

o *Output:* Public/secret key pair (Pk_S, Sk_S) of the sender, where the first one is published and the second one will be secured by the sender.

- *KeyGen$_R$:* Performed by the data receiver.

o *Input: Params.*
o *Process:*

i. **Chooses $y \in_R \mathbb{Z}_p$ as the receiver's secret key Sk_R and computes his public key $Pk_R = h^y$.**

o *Output:* Public/secret key pair (Pk_R, Sk_R) of the receiver, where the first one is published and the second one will be secured by the receiver.

- *PEAKS:* Performed by the data sender.

o *Input:* A keyword w, a data receiver's public key Pk_R, and a data sender's secret key Sk_S.
o *Process:*

i. Chooses $r \in_R \mathbb{Z}_p$ and computes $c_w^1 = H(w)^{Sk_S} \cdot g^r$ and $c_w^2 = (Pk_R)^r$.

o *Output:* $C_w = (c_w^1, c_w^2)$ as the cipher-keyword corresponding to keyword w.

- *Trapdoor:* Performed by the data receiver.

o *Input:* A keyword w, a data sender's public key Pk_S, and a data receiver's secret key Sk_R.
o *Process:*

i. Computes $T_w = e(H(w)^{Sk_R}, Pk_S)$.

o *Output:* T_w as the trapdoor corresponding to keyword w.

- *Test:* Performed by the server.

o *Input:* A trapdoor T_w, a cipher-keyword $C_{w'} = (c_{w'}^1, c_{w'}^2)$, the data sender's public key Pk_S, and the data receiver's public key Pk_R.
o *Process:*

i. Verifies $e(c_{w'}^1, Pk_R) =^? T_w \cdot e(g, c_{w'}^2)$.

o *Output:* 1 if the above equation holds, and 0 otherwise.

6 Analysis of the improved scheme

6.1 Security analysis

In this section, we analyse the security of our proposed scheme and show that under certain intractability assumptions, there exists no polynomial-time adversary which could distinguish trapdoors or cipher-keywords (as explained in Section 3). The following theorem formally states this fact. Our proof is essentially adapted from [14] along with applying the modifications brought forth in our proposed security model.

Theorem 2: The PAEKS scheme proposed in Section 5 is secure in the sense of Definition 5 in the random oracle model assuming DBDH problem and mDLIN problem are intractable.

Proof: To prove the theorem, we should show that the advantage of any polynomial-time adversary playing Game 1 or Game 2 (defined in Section 3) is negligible. In other words, we should prove the following two claims.

Claim 1: For any polynomial-time adversary $\mathscr{A}$, $\mathrm{Adv}_{\mathscr{A}}^T(\lambda)$ is negligible.

Proof: Let $\mathscr{A}$ be a polynomial-time adversary that has advantage ε in breaking trapdoor indistinguishability, i.e. $\mathrm{Adv}_{\mathscr{A}}^T(\lambda) = \epsilon$. We show that a polynomial-time algorithm $\mathscr{B}$ can use $\mathscr{A}$ and solve the DBDH problem. Suppose that $(G_1, G_T, e, p, g, g^x, g^y, g^z, Z)$ is given to $\mathscr{B}$ as the input of a DBDH problem. Note that x, y, z are randomly chosen from $\mathbb{Z}_p$, and Z is either equal to $e(g,g)^{xyz}$ (in this case let $b = 0$) or equal to a random element of G_T (in this case let $b = 1$). Algorithm $\mathscr{B}$ simulates the challenger of Game 2 and interacts with adversary $\mathscr{A}$ as follows:

- *Initialisation:* $\mathscr{B}$ chooses $\alpha \in_R \mathbb{Z}_p$ and sets $h = g^\alpha$, Param $= (G_1, G_T, e, p, g, h)$, $Pk_S = (g^x)^\alpha$, and $Pk_R = (g^y)^\alpha$ (i.e. $Pk_S = h^x$ and $Pk_R = h^y$). Then, it gives $(\mathrm{Param}, Pk_S, Pk_R)$ to $\mathscr{A}$.
- *Phase 1:* $\mathscr{B}$ provides the following oracles access for $\mathscr{A}$:

o $\mathscr{O}_H(w_i)$. $\mathscr{B}$ maintains a list L_H to respond hash queries which are initially empty. Given a keyword w_i, if w_i does not exist in L_H, $\mathscr{B}$ selects $a_i \in_R \mathbb{Z}_p$. Then, by tossing a biased coin c_i such that $\Pr[c_i = 0] = \delta$ (δ will be determined later), $\mathscr{B}$ sets $h_i = g^z \cdot g^{a_i}$ if $c_i = 0$, and sets $h_i = g^{a_i}$ otherwise. The tuple (w_i, h_i, a_i, c_i) is added into L_H. $H(w_i) = h_i$ is output as the hash value of w_i.
o $\mathscr{O}_{\mathrm{PAEKS}}(w_i, Pk)$. Given a keyword w_i and a public key Pk, $\mathscr{B}$ retrieves (w_i, h_i, a_i, c_i) by running $\mathscr{O}_H(w_i)$. If $c_i = 0$, it aborts and outputs a random bit b' as its guess of b. Otherwise, $\mathscr{B}$ selects $r_i \in_R \mathbb{Z}_p$ and computes the cipher-keyword $C_{w_i} = (C_{w_i}^1, C_{w_i}^2) = ((g^x)^{a_i} \cdot g^{r_i}, (Pk)^{r_i})$. C_{w_i} which is equal to PAEKS (w_i, Sk_S, Pk) is output by the oracle.
o $\mathscr{O}_{\mathrm{Trapdoor}}(w_i, Pk)$. Given a keyword w_i and a public key Pk, $\mathscr{B}$ retrieves (w_i, h_i, a_i, c_i) by running $\mathscr{O}_H(w_i)$. If $c_i = 0$, it aborts and outputs a random bit b' as its guess of b. Otherwise, the trapdoor $T_{w_i} = e((g^y)^{a_i}, Pk)$ is computed and output by the oracle. In this way, $T_{w_i} = \mathrm{Trapdoor}(w_i, Pk, Sk_R)$.

- *Challenge:* Eventually $\mathscr{A}$ outputs a pair of keywords (w_0^*, w_1^*) (with the restriction that $\mathscr{O}_{\mathrm{PAEKS}}(w_i^*, Pk_R)$ and $\mathscr{O}_{\mathrm{Trapdoor}}(w_i^*, Pk_S)$ for $i = 0, 1$ have never been queried in phase 1). Algorithm $\mathscr{B}$ generates the challenge trapdoor as follows: he retrieves $(w_i^*, h_i^*, a_i^*, c_i^*)$ by running $\mathscr{O}_H(w_i^*)$ for $i = 0, 1$. If $c_0^* = c_1^* = 1$, $\mathscr{B}$ aborts and outputs a random bit b' as its guess of b. Otherwise, if $c_0^* = 0$ or $c_1^* = 0$, let $\hat{b}$ be the bit such that $c_{\hat{b}}^* = 0$. $\mathscr{B}$ computes the trapdoor $T_{w_{\hat{b}}^*} = e((g^y)^{a_{\hat{b}}^*}, h^x) \cdot Z^\alpha$. Note that if $Z = e(g,g)^{xyz}$, then $T_{w_{\hat{b}}^*} = e(g,h)^{xy(z + a_{\hat{b}}^*)} = e((h_{\hat{b}}^*)^y, h^x)$ and if Z is random, then $T_{w_{\hat{b}}^*}$ is random too. $\mathscr{B}$ gives $T_{w_{\hat{b}}^*}$ to $\mathscr{A}$.
- *Phase 2:* $\mathscr{A}$ can continue to access the oracles in phase 1 with the restriction that $\mathscr{O}_{\mathrm{PAEKS}}(w_i^*, Pk_R)$ and $\mathscr{O}_{\mathrm{Trapdoor}}(w_i^*, Pk_S)$ should never be queried for $i = 0, 1$. $\mathscr{B}$ responds to these queries as before.
- *Response:* Eventually, $\mathscr{A}$ outputs its guess $\hat{b}'$. If $\hat{b}' = \hat{b}$, $\mathscr{B}$ outputs $b' = 0$; otherwise, it outputs $b' = 1$.

This completes the description of algorithm $\mathscr{B}$. First, we analyse the probability that $\mathscr{B}$ does not abort during the simulation. Suppose that the adversary issues at most q_C and q_T queries to the cipher-keyword oracle $\mathscr{O}_{\text{PAEKS}}$ and the trapdoor oracle $\mathscr{O}_{\text{Trapdoor}}$, respectively. The probability that $\mathscr{B}$ does not abort as a result of any of $\mathscr{A}$'s queries is $(1-\delta)^{q_T+q_C}$. Moreover, the probability that $\mathscr{B}$ does not abort during the challenge phase is $1-(1-\delta)^2$. Let Abort denote the event that $\mathscr{B}$ aborts during the game. Therefore, $\Pr[\overline{\text{Abort}}] = (1-\delta)^{q_T+q_C}\cdot(1-(1-\delta)^2)$. This probability takes the maximum value

$$\left(\frac{q_T+q_C}{q_T+q_C+2}\right)^{(q_T+q_C)/2}\frac{2}{q_T+q_C+2}\simeq\frac{2}{(q_T+q_C)e},$$

when

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

which is non-negligible.

Now, we compute the probability that $\mathscr{B}$ succeeds in solving the DBDH problem: $\Pr[b'=b] = \Pr[b'=b\wedge\text{Abort}] + \Pr[b'=b\wedge\overline{\text{Abort}}] = \Pr[b'=b|\text{Abort}]\Pr[\text{Abort}] + \Pr[b'=b|\overline{\text{Abort}}]\Pr[\overline{\text{Abort}}] = (1/2)(1-\Pr[\overline{\text{Abort}}]) + (\epsilon+(1/2))\cdot\Pr[\overline{\text{Abort}}] = (1/2)+\epsilon\cdot\Pr[\overline{\text{Abort}}]$. Therefore, the DBDH assumption in G_1 implies that $\epsilon(=\text{Adv}^T_{\mathscr{A}}(\lambda))$ is a negligible function in the security parameter and the proof of this claim is completed.□

Claim 2: For any polynomial-time adversary $\mathscr{A}$, $\text{Adv}^C_{\mathscr{A}}(\lambda)$ is negligible.

Proof: Let $\mathscr{A}$ be a polynomial-time adversary that has advantage ϵ' in breaking cipher-keyword indistinguishability, i.e. $\text{Adv}^C_{\mathscr{A}}(\lambda)=\epsilon'$. We show that a polynomial-time algorithm $\mathscr{B}$ can use $\mathscr{A}$ and solve the mDLIN problem.

Suppose that $(G_1, G_T, e, p, g, g^x, g^y, g^{ry}, g^{s/x}, Z)$ is given to $\mathscr{B}$ as the input of a mDLIN problem. Note that x, y, r, s are randomly chosen from $\mathbb{Z}_p$, and Z is either equal to g^{r+s} (in this case let $b=0$) or equal to a random element of G_1 (in this case let $b=1$). Algorithm $\mathscr{B}$ simulates the challenger of Game 1 and interacts with adversary $\mathscr{A}$ as follows:

- *Initialisation:* $\mathscr{B}$ chooses $\alpha\in_R\mathbb{Z}_p$ and sets $h=g^\alpha$, $\text{Param}=(G_1, G_T, e, p, g, h)$, $Pk_S=(g^x)^\alpha$, and $Pk_R=(g^y)^\alpha$ (i.e. $Pk_S=h^x$ and $Pk_R=h^y$). Then, it gives $(\text{Param}, Pk_S, Pk_R)$ to $\mathscr{A}$.
- *Phase 1:* $\mathscr{B}$ provides the following oracles access for $\mathscr{A}$:

  ○ $\mathscr{O}_H(w_i)$. $\mathscr{B}$ maintains a list L_H to respond hash queries which are initially empty. Given a keyword w_i, if w_i does not exist in L_H, $\mathscr{B}$ selects $a_i\in_R\mathbb{Z}_p$. Then, by tossing a biased coin c_i such that $\Pr[c_i=0]=\delta$ (δ will be determined later), $\mathscr{B}$ sets $h_i=g^{s/x}\cdot g^{a_i}$ if $c_i=0$, and sets $h_i=g^{a_i}$ otherwise. The tuple (w_i, h_i, a_i, c_i) is added into L_H. $H(w_i)=h_i$ is output as the hash value of w_i.
  ○ $\mathscr{O}_{\text{PAEKS}}(w_i, Pk)$. $\mathscr{B}$ answers to this query the same as explained in the proof of Claim 1.
  ○ $\mathscr{O}_{\text{Trapdoor}}(w_i, Pk)$. $\mathscr{B}$ answers to this query the same as explained in the proof of Claim 1.
- *Challenge:* Eventually $\mathscr{A}$ outputs a pair of keywords (w_0^*, w_1^*) (with the restriction that $\mathscr{O}_{\text{PAEKS}}(w_i^*, Pk_R)$ and $\mathscr{O}_{\text{Trapdoor}}(w_i^*, Pk_S)$ for $i=0,1$ have never been queried in phase 1). Algorithm $\mathscr{B}$ generates the challenge cipher-keyword as follows: He retrieves (w_i^*, h_i, a_i, c_i) by running $\mathscr{O}_H(w_i)$ for $i=0,1$. If $c_0^*=c_1^*=1$, $\mathscr{B}$ aborts and outputs a random bit b' as its guess of b. Otherwise, if $c_0^*=0$ or $c_1^*=0$, let $\hat{b}$ be the bit such that $c_{\hat{b}}^*=0$. By selecting $r\in_R\mathbb{Z}_p$, $\mathscr{B}$ computes the cipher-keyword $C_{w_{\hat{b}}^*}=(C^1_{w_{\hat{b}}^*}, C^2_{w_{\hat{b}}^*})=(Z\cdot g^{a_{\hat{b}}^*}\cdot(g^x)^{a_{\hat{b}}^*}, (h^y)^r\cdot(h^y)^{a_{\hat{b}}^*})$. Note that if $Z=g^{r+s}$, then $C_{w_{\hat{b}}^*}=(C^1_{w_{\hat{b}}^*}, C^2_{w_{\hat{b}}^*})=(g^{(r+a_{\hat{b}}^*)+(s+x\cdot a_{\hat{b}}^*)}, h^{y\cdot(r+a_{\hat{b}}^*)})=((h_{\hat{b}}^*)^x\cdot g^{r+a_{\hat{b}}^*}, h^{y\cdot(r+a_{\hat{b}}^*)})$ and if Z is random, $C_{w_{\hat{b}}^*}$ is random too. $\mathscr{B}$ gives $C_{w_{\hat{b}}^*}$ to $\mathscr{A}$.
- *Phase 2:* $\mathscr{A}$ can continue to access the oracles in phase 1 with the restriction that $\mathscr{O}_{\text{PAEKS}}(w_i^*, Pk_R)$ and $\mathscr{O}_{\text{Trapdoor}}(w_i^*, Pk_S)$ should never be queried for $i=0,1$. $\mathscr{B}$ responds to these queries as before.
- *Response:* Eventually, $\mathscr{A}$ outputs its guess $\hat{b}'$. If $\hat{b}'=\hat{b}$, $\mathscr{B}$ outputs $b'=0$; otherwise, it outputs $b'=1$.

This completes the description of algorithm $\mathscr{B}$. Let *Abort* denote the event that $\mathscr{B}$ aborts during the game. The same as computed in the proof of Claim 1, $\Pr[\overline{\text{Abort}}]=(1-\delta)^{q_T+q_C}\cdot(1-(1-\delta)^2)$. This probability takes the maximum value

$$\left(\frac{q_T+q_C}{q_T+q_C+2}\right)^{(q_T+q_C)/2}\frac{2}{q_T+q_C+2}\simeq\frac{2}{(q_T+q_C)e},$$

when

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

which is non-negligible.

Now, we compute the probability that $\mathscr{B}$ succeeds in solving the mDLIN problem:

$$\begin{aligned}\Pr[b'=b] &= \Pr[b'=b\wedge\text{Abort}]+\Pr[b'=b\wedge\overline{\text{Abort}}]\\ &= \Pr[b'=b|\text{Abort}]\Pr[\text{Abort}]\\ &\quad+\Pr[b'=b|\overline{\text{Abort}}]\Pr[\overline{\text{Abort}}]\\ &= \frac{1}{2}(1-\Pr[\overline{\text{Abort}}])+\left(\epsilon'+\frac{1}{2}\right)\cdot\Pr[\overline{\text{Abort}}]\\ &= \frac{1}{2}+\epsilon'\cdot\Pr[\overline{\text{Abort}}].\end{aligned}$$

Therefore, the mDLIN assumption implies that $\epsilon'(=\text{Adv}^C_{\mathscr{A}}(\lambda))$ is a negligible function in the security parameter and the proof of this claim is completed. □

6.2 Performance analysis

The results presented in HL-PAEKS show that their proposed scheme is almost as efficient as the PEKS of Boneh *et al.* [15] while adding the authentication property. Here, we made modifications to HL-PAEKS to overcome its short-comings in multi-user settings. The question naturally is whether some form of overhead appears in the process or not. The answer, fortunately, is that efficiency level remains untouched. The reason is that we solved the insecurity of their scheme in multi-user settings by utilising an additional generator for the employed group in global parameters. As a consequence, the two schemes (i.e. ours and [14]) are the same in terms of communication and computation costs. Therefore, we use the corresponding tables from [14] in order to have a comparison between some related schemes in [14] and ours. Although these tables are the same as what already exists in [14], we decided to include them here for the sake of completeness. Table 1 shows the computation costs of the algorithms that generate cipher-keywords and trapdoors and the test algorithm, where E, H, and P denote the evaluation of a modular exponentiation, a collision-resistant hash function, and a bilinear

Table 1 Computation efficiency comparison

Scheme	PEKS or PAEKS	Trapdoor	Test
[15]	$2E + 2H + P$	$E + H$	$H + P$
[17]	$2E + H + P$	$2E + H$	$E + P$
[18]	$3E + 4H + P$	$E + H$	$H + P$
[19]	$3E + 2H + 2P$	$E + H$	$H + P$
[20]	$6E + 2H + P$	$2E + H$	$E + H + P$
[21]	$6E + H + 3P$	$2E + H$	$E + H + P$
[22]	$2E + 2H + 9P$	$H + P$	$E + H + P$
[9]	$2E + 2H + P$	$3E + 2H$	$E + H + P$
[23]	$2E + 2H + 3P$	$2E + H$	$E + H + P$
[24]	$9E + 3H + 3P$	$2E$	$5E + H + 4P$
[14]	$3E + H$	$E + H + P$	$2P$
ours	$3E + H$	$E + H + P$	$2P$

Table 2 Communication efficiency comparison

Scheme	$\|Pk\|$	$\|C_w\|$	$\|T_w\|$
[15]	$\|G_1\|$	$\|G_1\| + n$	$\|G_1\|$
[17]	$2\|G_1\|$	$2\|G_1\| + \|G_T\|$	$\|G_1\| + \|\mathbb{Z}_p\| + n$
[18]	$\|G_1\|$	$\|G_1\| + 3n$	$\|G_1\|$
[19]	$2\|G_1\|$	$\|G_1\| + n$	$\|G_1\|$
[20]	$\|G_1\|$	$2\|G_1\| + n$	$\|G_1\|$
[21]	$2\|G_1\|$	$2\|G_1\| + 2\|G_T\|$	$\|G_1\| + \|\mathbb{Z}_p\|$
[22]	$2\|G_1\|$	$\|G_1\| + n$	$2\|G_1\|$
[9]	$2\|G_1\|$	$\|G_1\| + n$	$2\|G_1\|$
[23]	$3\|G_1\|$	$\|G_1\| + n$	$2\|G_1\|$
[24]	$2\|G_1\|$	$5\|G_1\| + 3\|G_T\|$	$3\|G_1\|$
[14]	$2\|G_1\|$	$\|G_1\|$	$\|G_T\|$
ours	$2\|G_1\|$	$\|G_1\|$	$\|G_T\|$

pairing, respectively. Table 2 is dedicated to the communication cost in terms of the sizes of the public key, cipher-keyword and trapdoor, where $|G_1|$, $|G_T|$, and $|\mathbb{Z}_p|$ denote the length of an element in group G_1, G_T, and $\mathbb{Z}_p$, respectively, and n is length of the security parameter.

As the computations of the two schemes are the same in all algorithms, our improved scheme and the scheme in [14] possess the same experimental results. Hence, the interested readers can refer to [14] in order to observe the corresponding running times.

7 Conclusion

To withstand inside KGA, Huang and Li introduced the notion of PAEKS and defined its security model in 2017. They also presented a concrete PAEKS scheme secure in their model. In this study, we first show that the presented security model has a vital deficiency that renders it inapplicable in practice. We also prove that the PAEKS scheme of Huang and Li is not secure against inside/outside KGA either. We then proceed to redefine the security model to overcome the problem and provide improvements to fix the flaw in the concrete construction. Our scheme is shown to be secure in the improved security model as well.

8 References

[1] Song, D.X., Wagner, D., Perrig, A.: 'Practical techniques for searches on encrypted data'. Proc. 2000 IEEE Symp. on Security and Privacy (S&P 2000), Berkeley, CA, USA, 2000, pp. 44–55

[2] Curtmola, R., Garay, J., Kamara, S., *et al.*: 'Searchable symmetric encryption: improved definitions and efficient constructions', *J. Comput. Secur.*, 2011, **19**, (5), pp. 895–934

[3] Tang, Q.: 'Nothing is for free: security in searching shared and encrypted data', *IEEE Trans. Inf. Forensics Sec.*, 2014, **9**, (11), pp. 1943–1952

[4] Asharov, G., Naor, M., Segev, G., *et al.*: 'Searchable symmetric encryption: optimal locality in linear space via two-dimensional balanced allocations'. Proc. 48th Annual ACM Symp. on Theory of Computing (STOC'16), Cambridge, MA, USA, 2016, pp. 1101–1114

[5] Wang, B., Li, M., Wang, H.: 'Geometric range search on encrypted spatial data', *IEEE Trans. Inf. Forensics Sec.*, 2016, **11**, (4), pp. 704–719

[6] Byun, J.W., Rhee, H.S., Park, H.A., *et al.*: 'Off-line keyword guessing attacks on recent keyword search schemes over encrypted data'. Workshop on Secure Data Management, Seoul, Korea, 2006, pp. 75–83

[7] Fang, L., Susilo, W., Ge, C., *et al.*: 'Public key encryption with keyword search secure against keyword guessing attacks without random oracle', *Inf. Sci.*, 2013, **238**, pp. 221–241

[8] Pakniat, N.: 'Public key encryption with keyword search and keyword guessing attack: a survey'. Proc. 13th Int. Iranian Society of Cryptology Conf. on Information Security and Cryptology (ISCISC), Tehran, Iran, 2016, pp. 1–4

[9] Rhee, H.S., Park, J.H., Susilo, W., *et al.*: 'Trapdoor security in a searchable public-key encryption scheme with a designated tester', *J. Syst. Softw.*, 2010, **83**, (5), pp. 763–771

[10] Guo, L., Yau, W.C.: 'Efficient secure-channel free public key encryption with keyword search for EMRs in cloud storage', *J. Med. Syst.*, 2015, **39**, (2), pp. 1–11

[11] Tang, Q., Chen, L.: 'Public-key encryption with registered keyword search'. European Public Key Infrastructure Workshop, Pisa, Italy, 2009, pp. 163–178

[12] Bösch, C., Tang, Q., Hartel, P., *et al.*: 'Selective document retrieval from encrypted database'. Int. Conf. on Information Security, Germany, 2012, pp. 224–241

[13] Xu, P., Jin, H., Wu, Q., *et al.*: 'Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack', *IEEE Trans. Comput.*, 2013, **62**, (11), pp. 2266–2277

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

[15] Boneh, D., Crescenzo, G.D., Ostrovsky, R., *et al.*: 'Public key encryption with keyword search'. Proc. Eurocrypt, Interlaken, Switzerland, 2004 (LNCS, 3027), pp. 506–522

[16] Baek, J., Steinfeld, R., Zheng, Y.: 'Formal proofs for the security of signcryption', *J. Cryptol.*, 2007, **20**, (2), pp. 203–235

[17] Park, D.J., Kim, K., Lee, P.J.: 'Public key encryption with conjunctive field keyword search'. Information Security Applications. WISA 2004, Jeju Island, Korea, 2005 (LNCS, 3325), pp. 73–86

[18] Baek, J., Safavi-Naini, R., Susilo, W.: 'On the integration of public key data encryption and public key encryption with keyword search'. Information Security (ISC 2006), Samos Island, Greece, 2006 (LNCS, **7176**), pp. 217–232

[19] Baek, J., Safavi-Naini, R., Susilo, W.: 'Public key encryption with keyword search revisited'. Computational Science and Its Applications (ICCSA 2008), Perugia, Italy 2008 (LNCS, **5072**), pp. 1249–1259

[20] Gu, C., Zhu, Y., Pan, H.: 'Efficient public key encryption with keyword search schemes from pairings'. Computational Science and Its Applications (ICCSA 2008), Xining, China, 2008 (LNCS, **5072**), pp. 372–383

[21] Fang, L., Susilo, W., Ge, C., *et al.*: 'A secure channel free public key encryption with keyword search scheme without random oracle'. Cryptology and Network Security (CANS 2009), Kanazawa, Japan, 2009 (LNCS, **5888**), pp. 248–258

[22] Rhee, H.S., Susilo, W., Kim, H.J.: 'Secure searchable public key encryption scheme against keyword guessing attacks', *IEICE Electron. Express*, 2009, **6**, (5), pp. 237–243

[23] Hu, C., Liu, P.: 'A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension'. Advances in Computer Science, Environment, Ecoinformatics, and Education (CSEE 2011), Wuhan, China, 2011 (Communications in Computer and Information Science, **215**), pp. 131–136

[24] Shao, Z.Y., Yang, B.: 'On security against the server in designated tester public key encryption with keyword search', *Inf. Process. Lett.*, 2015, **115**, (12), pp. 957–961