International Journal of Cyber-Security and Digital Forensics (IJCSDF) 9(2): 90-101
The Society of Digital Information and Wireless Communications (SDIWC), 2020 ISSN: 2305-0011

A Secure and Efficient Public Key Authenticated Encryption with Multi-keywords Search Scheme against Inside Keyword Guessing Attack

Yang Ma and Hassan Kazemian
School of Computing and Digital Media, London Metropolitan University
166-220 Holloway Road, London (N7 8DB).
yam0209@my.londonmet.ac.uk and kazemian@staff.londonmet.ac.uk

ABSTRACT

How to securely and efficiently search encrypted messages over the networked servers becomes a thorny problem. To solve it, Public Key Encryption with Keyword Search (PEKS) was firstly introduced in 2004. Since then, PEKS has witnessed a huge development and has expanded with more functionality and greater security. Many current PEKS schemes could prevent Off-line Keyword Guessing Attack (OKGA). But almost all PEKS schemes are vulnerable to Inside Keyword Guessing Attack (IKGA). This paper gives a definition of *Public Key Authenticated Encryption with Multi-keywords Search (PAEMKS)* which then subsequently presents a concrete construction of *PAEMKS*. The proposed scheme has the properties of Ciphertext Indistinguishability (CI) and Trapdoor Indistinguishability (TI) and incorporates with User Authentication technique, therefore, it is secure against both OKGA and IKGA. Besides, *PAEMKS* solves Multiple Keywords Search problem.

KEYWORDS

Public Key Encryption with Keyword Search (PEKS), Off-line Keyword Guessing Attack (OKGA), Inside Keyword Guessing Attack (IKGA), Ciphertext Indistinguishability (CI), Trapdoor Indistinguishability (TI), User Authentication, Multiple Keywords Search.

1 INTRODUCTION

Recently, many companies or people upload their data into the cloud servers [1, 2] to save huge human and material resources. However, storing data into the remote servers may lead in some negative effects. For example, the attackers could intercept or even manipulate the data packages on the public networks. In addition, the adversaries could exploit the physical leakage, such as power consumption and electromagnetic emission, to disclose secrets during the processing of a cryptographic operation by side channel analysis. To ensure data transmission security, the cryptographic operations must be applied to it. Interestingly, PEKS protects data transmission security well. More specially, PEKS enables users to search encrypted documents by a particular keyword without compromising the original data security.

Boneh, Di Crescenzo, Ostrovsky and Persiano (BDOP) revisited the Identity Based Encryption (IBE) and then defined the first PEKS scheme [3] in 2004. Although the proposed PEKS scheme satisfied Indistinguishability under Chosen Plaintext Attack (IND-CPA) secure, it brought in some weaknesses. For instance, the secure channel (i.e. SSL) must be built between the receiver and the online server. Besides, Byun et al. [4] pointed out that PEKS was vulnerable to OKGA.

There is no doubt that establishing a secure channel consumes plenty of resources and seems impossible in some cases. Therefore, Baek et al. [5] introduced a new PEKS scheme in 2008 removing the secure channel from the BDOP's PEKS scheme. However, the server is often assumed to be honest-but-curious and therefore, it may exploit the information related to the keyword. In the same year, Yau et al. [6] proposed that SCF-PEKS scheme was vulnerable to OKGA. Jeong et al. [7] then came up with an open problem: Is building a secure and consistent PEKS schemes against OKGA possible? Later on, Tang et al. [8] introduced a

PEKS scheme to resist OKGA, but the encryption algorithm was complex. Thereafter, Rhee et al. [9] incorporated with the TI to SCF-PEKS scheme in order to prevent OKGA. In 2013, Zhao et al. [10] came up with a new efficient SCF-PEKS scheme that also satisfied TI and CI to prevent OKGA. However, Yau et al. [11] found that the dPEKS was vulnerable to On-line KGA. Latterly, Chen firstly analyzed the drawback of dPEKS and proposed an extension of dPEKS (called SPEKS) [12] to solve both Off-line and On-line KGA. But the SPEKS required double encryptions and therefore, it consumed high computational costs. In 2019, Noroozi and Eslami proposed an efficient PEKS scheme by applying a re-randomization technique and the new proposed scheme was proved to be semantically secure against both Off-line and On-line KGA [13].

The PEKS schemes above only tolerate "exact" keyword search rather than solving format error ("P.S" and "P.S.") and/or spell error ("budget" and "buddget"). In 2010, Li et al. [14] came up with the definition of "Fuzzy Keyword Search" to PEKS scheme to solve format error and/or spelling inconsistency problems. But the proposed scheme suffered OKGA. Afterwards, Xu et al. [15] presented a sufficient PEKS scheme with Fuzzy Keyword Search to resist OKGA.

Many PEKS mechanisms above are able to solve Single Keyword Search issues, instead of Multiple Keywords Search problems. *"Public Key Encryption with Multi-keywords Search (MPEKS)"* [5] was defined by Baek et al. in 2008 to solve the Multiple Keywords Search issue. But a secure channel should be built for delivering trapdoor queries. In 2016, Wang et al. [16] came up with a SCF-MPEKS model removing the secure channel. However, SCF-MPEKS may suffer OKGA, if the malicious server or receiver published its private key to the outside networks. In 2018, Ma and Kazemian [17] introduced a new Trapdoor-indistinguishable SCF-MPEKS scheme to prevent OKGA.

If the inside adversary executes one extra bilinear pairing operation, many current PEKS schemes will suffer IKGA. Therefore, Li et al. [18] incorporated with a session key establishment to PEKS scheme, which was the first attempt to resist IKGA. In 2018, Noroozi et al. [19] revisited Li et al.'s PEKS scheme and pointed out that the inside attacker could still launch attacks to search the corresponding keywords to any PEKS encryption. Then, they proposed a simple solution to fix the problem. Meanwhile, Huang and Li [20] proposed a new PAEKS scheme applying User Authentication technique to resist IKGA. But PAEKS does not handle Multiple Keywords Search. In 2019, Noroozi and Eslami indicated that the PAEKS was inapplicable to the general public networks because it didn't support multi-user setting [21]. Besides, they found that the security for PAEKS was not sufficient and then refined the security models without any additional costs.

Apart from that, Zhang et al. [22] proposed a new PEKS scheme from lattice assumption in the standard model with quantum computers resistance, which was the milestone in the post-quantum cryptographic communication era. In 2020, Kazemian and Ma incorporated with Fuzzy Logic Technique to PEKS to solve Fuzzy Keyword Search problem (such as "latest") [23].

1.1 The Contributions

Firstly, the paper proposes a formal definition of *Public Key Authenticated Encryption with Multi-keywords Search (PAEMKS)* and then subsequently presents a concrete construction of PAEMKS. The proposed scheme solves both Single and Multiple Keyword(s) Search problems.

Besides, the security models for PAEMKS are also presented in this paper. More specially, the proposed scheme is proved to be semantically secure under the random oracle models with the Bilinear Diffie-Hellman (BDH) assumption so that it is secure against both OKGA and IKGA.

1.2 Paper Organization

The rest of the paper is made up of the following: some preliminaries are provided in Section 2. Section 3 revisits and analyses the scheme of *Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-keywords Search (tSCF-MPEKS)* [17]. Section 4 comes up with the Formal Definition, the Security Models, the Construction and Correctness of PAEMKS. Section 5 describes the security analysis of PAEMKS and also compares the performance and efficiency between the PAEMKS scheme and its counterparts. Finally, Conclusions will be illustrated in Section 6.

2 PRELIMINARIES

2.1 Mathematic Theories

2.1.1 Bilinear pairings

Let G_1 and G_T be two cyclic groups. G_1 is an additive group and G_T is a multiplicative group respectively. The prime number P is the generator of G_1 while g is the order of G_1 respectively. Also, let a and b be the elements of Z_P. A bilinear pairing can be mapped to $e : G_1 \times G_1 \rightarrow G_T$, which satisfies the following properties:

i. Bilinearity: $e(aU, bV) = e(U, V)^{ab}$ for all $U, V \in G_1$ and $a, b \in Z_P$.

ii. Computability: $e(U, V) \in G_T$ is computable in a PPT algorithm, for any $U, V \in G_1$.

iii. Non-degenerate: $e(U, V) \neq 1$.

2.1.2 The Bilinear Diffie-Hellman (BDH) assumption [24]

Let P, aP, bP, cP be the inputs (where $a, b, c \in Z_P$), compute $e(P, P)^{abc} \in G_T$. An algorithm A will satisfy an advantage ε in G_1 to solve BDH assumption, if $Pr[A(P, aP, bP, cP) = e(P, P)^{abc}] \geq \varepsilon$. Therefore, if no PPT algorithm taking an advantage at least ε in G_1 solves BDH assumption, BDH assumption will hold in G_1

2.2 Overview of SCF-MPEKS Scheme

Sender, server and receiver are three main parties in SCF-MPEKS scheme. The sender generates a SCF-MPEKS ciphertext and the receiver generates a Trapdoor query. After that, the searchable ciphertext and the trapdoor query are transmitted to the server. Once the server receives these encrypted messages, it will call Test algorithm to match whether two encryptions have the same keyword or not.

SCF-MPEKS was proposed by Wang et al. [16], which contained six PPT algorithms below:

1. $KeyGen_P(1^n)$: Input 1^n and then produce a global parameter gp.
2. $KeyGen_{Ser}(gp)$: Input gp for creating the server's public and private keys (pk_S, sk_S).
3. $KeyGen_{Rec}(gp)$: Input gp for creating the receiver's public and private keys (pk_R, sk_R).
4. $SCF-MPEKS(pk_S, pk_R, W)$: Input pk_S and pk_R for creating a searchable ciphertext $S = SCF-MPEKS(pk_S, pk_R, W)$ of the multi-keywords, where $W = (w_1, w_2, \ldots, w_n)$.
5. $Trapdoor(sk_R, w)$: Input sk_R for creating a trapdoor query $T_w = Trapdoor(sk_R, w)$ of a keyword w.
6. $Test(T_w, sk_S, S)$: Input sk_S, $S = SCF-MPEKS(pk_S, pk_R, W)$ and $T_w = Trapdoor(sk_R, w)$. If W contains w, output "yes" and "no match", otherwise.

3 TRAPDOOR-INDISTINGUISHABLE SECURE CHANNEL FREE PUBLIC KEY ENCRYPTION WITH MULTI-KEYWORDS SEARCH

3.1 Formal Definition for tSCF-MPEKS

Ma and Kazemian found that SCF-MPEKS suffers OKGA and then proposed the *tSCF-MPEKS* [17] to fix this problem in Sep, 2018. The scheme consisted of six algorithms as follows:

1. $KeyGen_P(1^n)$: Input 1^n and then produce a global parameter gp.
2. $KeyGen_S(gp)$: Input gp for creating the server's public and private keys (pk_S, sk_S).
3. $KeyGen_{Rec}(gp)$: Input gp for creating the receiver's public and private keys (pk_R, sk_R).
4. $SCF-MPEKS(pk_S, pk_R, W)$: Input pk_S and pk_R for creating a searchable ciphertext

$S = SCF - MPEKS(pk_S, pk_R, W)$ of the multi-keywords, where $W = (w_1, w_2, \ldots, w_n)$.
5. $Trapdoor(pk_S, sk_R, w)$: Input pk_S and sk_R for creating a trapdoor query $T_w = Trapdoor(pk_S, sk_R, w)$ of a keyword w.
6. $Test(sk_S, S, T_w)$: Input sk_S, $S = SCF-MPEKS(pk_S, pk_R, W)$ and $T_w = Trapdoor(pk_S, sk_R, w)$. If W contains w, output "yes" and "no match", otherwise.

3.2 The Construction of tSCF-MPEKS

The concrete construction of tSCF-MPEKS is described in the following:
1. $KeyGen_P(1^n)$: Consider G_1 is an additive cyclic group and G_T is a multiplicative cyclic group. P ($P \geq 2^k$) and g are a random generator and an order of G_1 respectively. A bilinear pairing is a map $e: G_1 \times G_1 \rightarrow G_T$. Suppose $H: \{0,1\}^* \rightarrow G_1$ and $H^*: G_T \rightarrow \{0,1\}^*$ are two particular hash functions. The global parameter $gp = \{g, P, G_1, G_T, e, H, H^*\}$ is returned in the end.
2. $KeyGen_{Ser}(gp)$: The server chooses $x \in Z_P$ uniformly at random and then calculates $X = xP$. Besides, the server also chooses $K \in G_1$ uniformly at random. Therefore, the server's public and private keys are $pk_S = (gp, X, K)$ and $sk_S = (gp, x)$.
3. $KeyGen_{Rec}(gp)$: The receiver chooses $y \in Z_P$ uniformly at random and then calculates $Y = yP$. So, the receiver's public and private keys are $pk_R = (gp, Y)$ and $sk_R = (gp, y)$.
4. $SCF - MPEKS(pk_S, pk_R, W)$: The sender chooses $t \in Z_P$ uniformly at random and then calculates a searchable ciphertext $S = (A, B_1, B_2, \ldots, B_n) = [tX, H^*(Z_1), H^*(Z_2), \ldots, H^*(Z_n)]$, where $Z_1 = e(H(w_1), Y)^t$, $Z_2 = e(H(w_2), Y)^t, \ldots, Z_n = e(H(w_n), Y)^t$.
5. $Trapdoor(pk_S, sk_R, w)$: The receiver randomly chooses $t^* \in Z_P$ and then computes $T_w = (T_1, T_2)$, where $T_1 = yH(w^*) \oplus e(X, K)^{t^*+y}$ and $T_2 = e(X, t^*K)$.
6. $Test(sk_S, S, T_w)$: For $i \in \{1,2,...,n\}$, the server firstly computes $T = T_1 \oplus T_2 \bullet e(xK, Y) = yH(w^*)$. Then, the server checks if $H^*[e(T, \frac{A}{x})] = Z_i$. If so, output "yes" and "no match", otherwise.

3.3 Analysis of tSCF-MPEKS

The tSCF-MPEKS is vulnerable to the IKGA. More specially, the server is able to execute Test algorithm to search the relation between a keyword in ciphertext and a keyword in trapdoor. The details are introduced below.

Let the server be honest-but-curious. The server receives a Trapdoor $T_w = (T_1, T_2)$ from the receiver and then computes $T = T_1 \oplus T_2 \bullet e(xK, Y) = yH(w)$. Next, the server guesses an appropriate keyword w^* and calculates $H(w^*)$. After that, the server executes one bilinear pairing operation to check if $e(pk_R, H(w^*)) = e(P, T)$. The equation holds for $w^* = w$ as $e(pk_R, H(w^*)) = e(yP, H(w^*)) = e(P, yH(w)) = e(P, T)$. Overall, the malicious server could guess the correct keyword by trial and error and therefore, tSCF-MPEKS is not able to resist the IKGA.

4 PUBLIC KEY AUTHENTICATED ENCRYPTION WITH MULTI-KEYWORDS SEARCH

4.1 Formal Definition for PAEMKS

In order to resist IKGA, the server should not encrypt a keyword by itself. Huang and Li [20] pointed out that keyword authentication is a necessary step in PEKS algorithm. Therefore, the malicious server is not able to launch Keyword Guessing Attack to find the relation between a keyword in ciphertext and a keyword in trapdoor.

The new PAEMKS scheme below not only resists IKGA but also enables users to search encrypted files by the multiple keywords. The details are introduced in the following.
1. $KeyGen_P(1^n)$: Input 1^n and then produce a global parameter gp.
2. $KeyGen_{Sen}(gp)$: Input gp for creating the sender's public and private key (pk_{Sen}, sk_{Sen}).
3. $KeyGen_{Rec}(gp)$: Input gp for creating the receiver's public and private key (pk_{Rec}, sk_{Rec}).
4. $SCF - MPEKS(sk_{Sen}, pk_{Rec}, W)$: Input sk_{Sen} and pk_{Rec} for creating a searchable ciphertext

$S = SCF - MPEKS(sk_{Sen}, pk_{Rec}, W)$, where the multiple keywords $W = (w_1, w_2, \ldots, w_n)$.

5. $Trapdoor(sk_{Rec}, pk_{Sen}, w)$: Input pk_{Sen} and sk_{Rec} for creating a trapdoor query $T_w = Trapdoor(sk_{Rec}, pk_{Sen}, w)$ of a keyword w.

6. $Test(T_w, S, pk_{Sen}, pk_{Rec})$: Input pk_{Sen}, pk_{Rec}, $S = SCF - MPEKS(sk_{Sen}, pk_{Rec}, W)$ and $T_w = Trapdoor(sk_{Rec}, pk_{Sen}, , w^*)$. If W contains w, output "yes" and "no match", otherwise.

4.2 Security Models for PAEMKS

As discussed in tSCF-MPEKS [17] and PAEKS [20], the proposed scheme is IND-CPA and Trapdoor-IND-CPA. The IND-CPA and Trapdoor-IND-CPA for PAEMKS scheme are formally defined in the following.

Suppose **A** and **E** are an attacker and a challenger respectively.

Game1: Ciphertext Indistinguishability

Setup: The challenger **E** calls $KeyGen_P(1^n)$, $KeyGen_{Sen}(gp)$ and $KeyGen_{Rec}(gp)$ to create the global parameter gp, sender's public and private keys (pk_{Sen}, sk_{Sen}) and receiver's public and private keys (pk_{Rec}, sk_{Rec}). After that, the attacker **A** obtains gp, pk_{Rec}, pk_{Sen} and keeps sk_{Rec}, sk_{Sen} from **A**.

Phase 1-1 (Queries): **A** sends the Trapdoor Oracle O_T and Ciphertext Oracle O_C as many times as possible to **E**.

Challenge: **A** deliveries a target keyword-vector pair (W_0, W_1) to **E** for all $W_0 = (w_{01}, w_{02}, \ldots, w_{0n})$ and $W_1 = (w_{11}, w_{12}, \ldots, w_{1n})$. Besides, W_0 and W_1 cannot be searched in **Phase 1-1**. Once **E** receives the target keyword-vector pair, he/she calls $SCF - MPEKS$ algorithm to create a searchable ciphertext $S = SCF - MPEKS(sk_{Sen}, pk_{Rec}, W_\lambda)$, where $\lambda \in \{0,1\}$. Finally, **A** will receive S from **E**.

Phase 1-2 (Queries): **A** asks **E** for Trapdoor Oracle O_T and Ciphertext Oracle O_C as many times as in **Phase 1-1**, only if $W \neq W_0, W_1$.

Guess: **A** outputs the guess $\lambda^* \in \{0,1\}$ and wins ***Game1***, if $\lambda^* = \lambda$.

The advantage of **A** winning ***Game1*** is in the following:

$$Adv^{IND-CPA}_{PAEMKS,A}(k) = |Pr[\lambda^* = \lambda] - 1/2| \quad (1)$$

Therefore, the PAEMKS model can be regarded as **IND-CPA** secure as long as the $Adv^{IND-CPA}_{PAEMKS,A}(k)$ is negligible.

Game2: Trapdoor Indistinguishability

Setup: The challenger **E** calls $KeyGen_P(1^n)$, $KeyGen_{Sen}(gp)$ and $KeyGen_{Rec}(gp)$ to create the global parameter gp, sender's public and private keys (pk_{Sen}, sk_{Sen}) and receiver's public and private keys (pk_{Rec}, sk_{Rec}). After that, the attacker **A** obtains gp, pk_{Rec}, pk_{Sen} and keeps sk_{Rec}, sk_{Sen} from A.

Phase 2-1 (Queries): **A** sends the Trapdoor Oracle O_T and Ciphertext Oracle O_C as many times as possible to **E**.

Challenge: **A** deliveries a target keyword pair (w_0, w_1) to **E**. Note that none of w_0 or w_1 has been queried in **Phase 2-1**. Once **E** receives the target keyword pair, he/she calls $Trapdoor$ algorithm to create a trapdoor query $T_w = Trapdoor(sk_{Rec}, pk_{Sen}, w_\lambda)$, where $\lambda \in \{0,1\}$. Finally, **A** will receive T_w from **E**.

Phase 2-2 (Queries): **A** asks **E** for Trapdoor Oracle O_T and Ciphertext Oracle O_C as many times as in **Phase 2-1**, only if $w \neq w_0, w_1$.

Guess: **A** outputs the guess $\lambda^* \in \{0,1\}$ and wins ***Game2***, if $\lambda^* = \lambda$.

The advantage of **A** winning ***Game2*** is in the following:

$$Adv^{Trap-IND-CPA}_{PAEMKS,A}(k) = |Pr[\lambda^* = \lambda] - 1/2| \quad (2)$$

Therefore, the PAEMKS model can be regarded as **Trapdoor-IND-CPA** secure as long as the $Adv^{Trap-IND-CPA}_{PAEMKS,A}(k)$ is negligible.

4.3 The Construction for PAEMKS

1. $KeyGen_P(1^n)$: Consider G_1 and G_T are additive cyclic group and multiplicative cyclic group. P $(P \geq 2^k)$ and g are a random generator and an order of G_1 respectively. A bilinear pairing is a map $e: G_1 \times G_1 \rightarrow G_T$. Suppose $H: \{0,1\}^* \rightarrow G_1$ is a particular hash function.

The global parameter $gp = \{g, P, G_1, G_T, e, H\}$ is returned in the end.

2. $KeyGen_{Sen}(gp)$: The sender randomly chooses $a \in Z_P$ and then computes $A = aP$. Therefore, the sender's public key is $pk_{Sen} = A$ and the private key is $sk_{Sen} = a$.

3. $KeyGen_{Rec}(gp)$: The receiver randomly chooses $b \in Z_P$ and then computes $B = bP$. Therefore, the receiver's public key is $pk_{Rec} = B$ and the private key is $sk_{Rec} = b$.

4. $SCF - MPEKS(sk_{Sen}, pk_{Rec}, W)$: The sender randomly chooses $t \in Z_P$ and $W = (w_1, w_2, \ldots, w_n)$. Then, he/she computes a searchable encryption $S = (L, N_1, N_2, \ldots, N_n) = [t \oplus pk_{Rec}, e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_1), pk_{Rec} \bullet t), e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_2), pk_{Rec} \bullet t), \ldots, e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_n), pk_{Rec} \bullet t)]$.

5. $Trapdoor(sk_{Rec}, pk_{Sen}, w)$: The receiver computes a trapdoor query $T_w = e(sk_{Rec} \bullet H(pk_{Sen}, pk_{Rec}, w), pk_{Sen})$.

6. $Test(T_w, S, pk_{Sen}, pk_{Rec})$: For $i \in \{1,2,...,n\}$, the server firstly calculates $L \oplus pk_{Rec}$ and then checks whether $T^t = N_i$ or not.

4.4 The Correctness for PAEMKS

Suppose W and w^* are a keyword-vector and keyword in $SCF - MPEKS$ and $Trapdoor$ algorithms respectively. The proposed scheme is completely correct, only if W contains w^*. The details are provided in the following.

The server initially calculates $L \oplus pk_{Rec} = t \oplus pk_{Rec} \oplus pk_{Rec} = t$. And then, the server checks whether $T^t = N_i$.

For $i \in \{1,2,...,n\}$,

$$\begin{aligned} T^t &= e(sk_{Rec} \bullet H(pk_{Sen}, pk_{Rec}, w^*), pk_{Sen})^t \\ &= e(b \bullet H(pk_{Sen}, pk_{Rec}, w^*), aP)^t \\ &= e(a \bullet H(pk_{Sen}, pk_{Rec}, w^*), bP)^t \\ &= e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w^*), pk_{Rec} \bullet t) \\ &= N_i \end{aligned}$$

Therefore, the algorithm is completely correct.

5 SECURITY AND PERFORMANCE

5.1 Security Analysis for PAEMKS

Game1: PAEMKS scheme is Ciphertext Indistinguishability under BDH assumption in G_1.

Proof: Assuming that the PPT **A** breaks the ciphertext privacy of PAEMKS with a non-negligible advantage ε_C. Suppose that **E** takes $(g, P, G_1, G_T, e, xP, yP, zP)$ as an input of BDH assumption whose running time is bounded by T. **E**'s aim is to calculate a BDH key $e(P, P)^{xyz}$ of xP, yP and zP using **A** 's IND-CPA.

Setup Simulation

E randomly chooses $x, y \in Z_P$ and then returns xP and yP as the public keys (pk_{Sen}, pk_{Rec}) of the sender and the receiver. Then, **E** generates a common parameter $cp = (g, P, G_1, G_T, e, H)$ and transmits (cp, pk_{Sen}, pk_{Rec}) to **A** .

Phase 1-1 Simulation (Queries)

For simplicity, three assumptions are introduced as follows.

1. **A** requests at most q_H, q_T, q_C to the hash function query O_H, the trapdoor query O_T and the ciphertext query O_C respectively.

2. **A** does not repeat any above queries.

3. **A** is not able to query (pk_{Sen}, w) to O_T nor (pk_{Rec}, w) to O_C before searching (pk_{Sen}, pk_{Rec}, w) to O_H.

The queries are simulated by **E** below.

For *Hash Query* O_H:

When **A** issues a query for a tuple $(pk_{Sen}, pk_{Rec}, w_i)$. To respond,

i. **E** randomly picks up a coin θ_i and then computes $Pr[\theta_i = 0] = \frac{1}{h+1}$.

ii. **E** randomly selects $f_i \in Z_P$. If $\theta_i = 0$, E will calculate $F_i = lP + f_iP$. If $\theta_i = 1$, E will calculate $F_i = f_iP$.

iii. **E** returns F_i as an answer to **A** and places $[(pk_{Sen}, pk_{Rec}, w_i), F_i, f_i, \theta_i]$ into H_List. The H_List is initially empty.

For *Trapdoor Query* O_T:

When **A** issues a query for the trapdoor corresponding to the keyword w. To respond, **E** computes $T_w = e(f_i \bullet pk_{Rec}, pk_{Sen}) = e(yF_1, pk_{Sen}) = e(sk_{Rec} \bullet H(pk_{Sen}, pk_{Rec}, w), pk_{Sen})$,

where T_w is a correct trapdoor under the sender's public key and the receiver's private key. After that, **E** returns T_w to **A**.

For *Ciphertext Query* O_C*:*

When **A** issues a query for the ciphertext corresponding to the keyword w. To respond, **E** randomly chooses $t \in Z_P$ and then computes $N_w = e(f_1 \bullet pk_{Sen}, pk_{Rec} \bullet t) = e(xF_1, pk_{Rec} \bullet t) = e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w), pk_{Rec} \bullet t)$, where N_w is a correct ciphertext under the sender's private key and the receiver's public key. After that, **E** returns N_w to **A**.

Challenge Simulation

A sends a keyword-vector pair (W_0^*, W_1^*) to **E**, where $W_0^* = (w_{01}, w_{02}, \ldots, w_{0n})$ and $W_1^* = (w_{11}, w_{12}, \ldots, w_{1n})$. It should be clear that (yP, w_0^*) and (yP, w_1^*) cannot be queried by O_T while (xP, W_0^*) and (xP, W_1^*) cannot be queried by O_C. Then, **E** returns a searchable ciphertext S in the following:

- **E** randomly selects $i \in \{1,2,...,n\}$.
- **E** calls the above algorithms to achieve two tuples $(w_{0i}^*, F_{0i}^*, f_{0i}^*, \theta_{0i}^*)$ and $(w_{1i}^*, F_{1i}^*, f_{1i}^*, \theta_{1i}^*)$. If $\theta_0 = \theta_1 = 1$, **E** will terminate the PAEMKS system and will output "Suspension".

Otherwise, **E** computes the ciphertext as follows:

- **E** recalls the above algorithms for simulating H function at $2(n-1)$ times to obtain two vectors of tuples $[(w_{01}^*, F_{01}^*, f_{01}^*, \theta_{01}^*), \ldots, (w_{0i-1}^*, F_{0i-1}^*, f_{0i-1}^*, \theta_{0i-1}^*), (w_{0i+1}^*, F_{0i+1}^*, f_{0i+1}^*, \theta_{0i+1}^*), \ldots, (w_{0n}^*, F_{0n}^*, f_{0n}^*, \theta_{0n}^*)]$ and $[(w_{11}^*, F_{11}^*, f_{11}^*, \theta_{11}^*), \ldots, (w_{1i-1}^*, F_{1i-1}^*, f_{1i-1}^*, \theta_{1i-1}^*), (w_{1i+1}^*, F_{1i+1}^*, f_{1i+1}^*, \theta_{1i+1}^*), \ldots, (w_{1n}^*, F_{1n}^*, f_{1n}^*, \theta_{1n}^*)]$. If $\theta_{0j}^* = \theta_{1j}^* = 0$ for all $j = 0,...,i-1,i+1,...,n$, **E** will terminate the PAEMKS system and will output "Suspension".

Otherwise, **E** returns as follows:

-- **E** randomly chooses $\beta \in \{0,1\}^d$.

-- **E** randomly chooses $J_j \in \{0,1\}^d$ and creates a target $SCF-MPEKS$ ciphertext $S^* = (L^*, N_1^*, N_2^*, \ldots, N_n^*)$.

So, E selects $t = z$. Then, **E** calculates

$S^* = (L^*, N_1^*, \ldots, N_{i-1}^*, N_{i+1}^*, \ldots, N_n^*) = [z \oplus yP, e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_1^*), pk_{Rec} \bullet t), \ldots, e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_{i-1}^*), pk_{Rec} \bullet t), e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_{i+1}^*), pk_{Rec} \bullet t), \ldots, e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_\beta^*), pk_{Rec} \bullet t)]$.

Note that $N_\beta = e(sk_{Sen} \bullet H(pk_{Sen}, pk_{Rec}, w_\beta^*), pk_{Rec} \bullet t) = e(xH(pk_{Sen}, pk_{Rec}, w_\beta^*), yP \bullet z) = e(xf_iP, yP \bullet z) = e(f_iP, P)^{xyz}$.

Phase 1-2 Simulation (Queries)

A asks **E** for any queries as in **Phase 1-1 Simulation (Queries)**, only if **A** does not issue $[(pk_{Sen}, w_0^*), (pk_{Sen}, w_1^*)]$ to O_T and $[(pk_{Rec}, W_0^*), (pk_{Rec}, W_1^*)]$ to O_C.

Guess

A outputs its guess $\beta^* \in \{0,1\}$. If $\beta = \beta^*$, **E** outputs "yes" and "no match", otherwise.

Analysis:

There are two events customising as follows:

$Event1$: **E** does not stop during the **Phase 1-1** and the **Phase 1-2**.

$Event2$: **E** does not stop during the **Challenge Simulation**.

Claim 1: $Pr[Event1] \geq (1 - \frac{1}{h+1})^{q_T+q_C}$ (3)

Consider that **A** does not issue the same keyword twice in O_T and O_C queries. So, $\frac{1}{h+1}$ is the probability causing **E** for suspension. According to the previous definition, **A** queries at most q_T trapdoor queries and q_C ciphertext queries, the probability that **E** does not terminate the PAEMKS system is at least $(1 - \frac{1}{h+1})^{q_T+q_C}$.

Claim 2: $Pr[Event2] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$ (4)

If $\theta_0 = \theta_1 = 1$, the PAEMKS system will be terminated by **E** during the **Challenge Simulation**. So, the probability that **E** does not suspend is $1 - (1 - \frac{1}{h+1})^2$. Besides, if $\theta_{0j}^* = \theta_{1j}^* = 0$ for all $j = 0,...,i-1,i+1,...,n$, **E** will also suspend here. Overall, the probability that **E** does not terminate the PAEMKS system during the **Challenge Simulation** is at least $(1 - \frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$.

Let $Event$ be an event that **E** does not suspend during the whole game. So, it is bounded by $Pr[Event] = Pr[Event1] \bullet Pr[Event2] = (1 - \frac{1}{h+1})^{q_T+q_C} \bullet (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$

If $h + 1 = q_T + q_C$, the $Pr[Event]$ will obtain the maximum value. Therefore,

$Pr[Event] = \frac{1}{e} \bullet (\frac{1}{q_T + q_C}) \bullet (\frac{q_T + q_C - 1}{q_T + q_C})^{2(n-1)}$, which is approximately equal to $\frac{1}{e(q_T + q_C)}$ and thus non-negligible.

Overall, the probability that **E** guessing the correct bit β is
$Pr[\beta' = \beta] = Pr[\beta' = \beta \wedge Pr[Event]] + Pr[\beta' = \beta \wedge Pr[\overline{Event}]] = Pr[\beta' = \beta \mid Pr[Event]]Pr[Event] + Pr[\beta' = \beta \mid Pr[\overline{Event}]]Pr[\overline{Event}] = \frac{1}{2} \bullet (1 - Pr[\overline{Event}]) + (\varepsilon_C + \frac{1}{2}) \bullet Pr[\overline{Event}] = \frac{1}{2} + \varepsilon_C \bullet Pr[\overline{Event}]$.
If ε_C is non-negligible, so is $|Pr[\beta' = \beta] - \frac{1}{2}|$.

Game2: PAEMKS scheme is Trapdoor Indistinguishability under BDH assumption in G_1.
Proof: Assuming that the PPT **A** breaks the trapdoor privacy of PAEMKS with a non-negligible advantage ε_T. Suppose that **E** takes $(g, P, G_1, G_T, e, xP, yP, zP)$ as an input of BDH assumption whose running time is bounded by T. **E**'s aim is to calculate a BDH key $e(P,P)^{xyz}$ of xP, yP and zP using **A** 's Trapdoor-IND-CPA.

Setup Simulation
E randomly chooses $x, y \in Z_P$ and then returns xP and yP as the public keys (pk_{Sen}, pk_{Rec}) of the sender and the receiver. Then, E generates the common parameter $cp = (g, P, G_1, G_T, e, H)$ and transmits (cp, pk_{Sen}, pk_{Rec}) to **A**.
Phase 2-1 Simulation (Queries)
For simplicity, three assumptions are introduced below.
1. **A** requests at most q_H, q_T, q_C to the hash function query O_H, the trapdoor query O_T and the ciphertext query O_C respectively.
2. **A** does not repeat any above queries.
3. **A** is not able to query (pk_{Sen}, w) to O_T nor (pk_{Rec}, w) to O_C before searching (pk_{Sen}, pk_{Rec}, w) to O_H.
The queries are simulated by **E** below.
For *Hash Query* O_H, *Trapdoor Query* O_T and *Ciphertext Query* O_C, **E** answers the same way as in the proof of **Game1**.
Challenge Simulation
A sends the keyword pair (w_0^*, w_1^*) to **E**. It should be clear that $[(yP, w_0^*), (yP, w_1^*)]$ cannot be queried by O_T and $[(xP, W_0^*), (xP, W_1^*)]$ cannot be queried by O_C. Then, **E** returns the challenge trapdoor T_w in the following:
- If $\theta_0 = \theta_1 = 1$, **E** will terminate the PAEMKS system and will output "Suspension".
Otherwise, **E** computes the trapdoor as follows:
- $T_\beta = M \bullet e(xP, yP)^{f_i}$. If $M = e(P,P)^{xyz}$, then $T_\beta = e(P,P)^{xy(z+f_i)} = e(F_i, (xy)P)$. If M is a random element of G_T, so is T_β.
Phase 2-2 Simulation (Queries)
A asks **E** for any queries as in **Phase 2-1 Simulation (Queries)**, only if **A** does not issue $[(pk_{Sen}, w_0^*), (pk_{Sen}, w_1^*)]$ to O_T and $[(pk_{Rec}, W_0^*)(pk_{Rec}, W_1^*)]$ to O_C.
Guess
A outputs its guess $\beta^* \in \{0,1\}$. If $\beta = \beta^*$, **E** outputs "yes" and "no match", otherwise.

Analysis
There are two events customising as follows:
$Event3$: **E** does not stop during the **Phase 2-1** and the **Phase 2-2**.
$Event4$: **E** does not stop during the **Challenge Simulation**.

Claim 3: $Pr[Event3] \geq (1 - \frac{1}{h+1})^{q_T + q_C}$ (5)
The proof of **Claim 3** is same as the proof of **Claim 1**, so it is omitted here.

Claim 4: $Pr[Event4] \geq 1 - (1 - \frac{1}{h+1})^2$ (6)
If $\theta_0 = \theta_1 = 1$, **E** will terminate the PAEMKS system during the **Challenge Simulation**. So, the probability that E does not suspend is $1 - (1 - \frac{1}{h+1})^2$.

Suppose $Event'$ is an event that **E** does not stop during the whole game. So, it is

bounded by $Pr[Event'] = Pr[Event3] \bullet Pr[Event4] = (1 - \frac{1}{h+1})^{q_T + q_C} \bullet (1 - (1 - \frac{1}{h+1})^2)$
If $\frac{1}{h+1} = 1 - \sqrt{\frac{q_T + q_C}{q_T + q_C + 2}}$, the $Pr[Event']$ will obtain the maximum value. Therefore,
$Pr[Event'] = (\frac{q_T + q_C}{q_T + q_C + 2})^{(q_T + q_C)/2} \bullet \frac{2}{q_T + q_C + 2}$,

which is approximately equal to $\frac{2}{e(q_T + q_C)}$ and thus non-negligible.

Overall, the probability that **E** guessing the correct bit β is
$Pr[\beta' = \beta] = Pr[\beta' = \beta \wedge Pr[Event']] + Pr[\beta' = \beta \wedge Pr[\overline{Event'}]] = Pr[\beta' = \beta \mid Pr[Event']]Pr[Event'] + Pr[\beta' = \beta \mid Pr[\overline{Event'}]]Pr[\overline{Event'}] = \frac{1}{2} \bullet (1 - Pr[\overline{Event'}]) + (\varepsilon_T + \frac{1}{2}) \bullet Pr[\overline{Event'}] = \frac{1}{2} + \varepsilon_T \bullet Pr[\overline{Event'}]$
If ε_T is non-negligible, so is $|Pr[\beta' = \beta] - \frac{1}{2}|$.

5.2 Performance and Efficiency for PAEMKS

This part presents a comparison of security between the proposed scheme (PAEMKS) and the other several PEKS schemes (PEKS[3], SCF-PEKS[5], dPEKS[9], PAEKS[20], MPEKS[5], SCF-MPEKS[16] and tSCF-MPEKS[17]). In addition, the performance of the proposed scheme is also described in this section.

TABLE 1. Comparison of Functionalities between PAEMKS and its counterparts

Scheme	CT Ind	Trap Ind	MS	IKGA
PEKS	Satisfied	Not satisfied	No	Suffered
SCF-PEKS	Satisfied	Not satisfied	No	Suffered
dPEKS	Satisfied	Satisfied	No	Suffered
PAEKS	Satisfied	Satisfied	No	Not suffered
MPEKS	Satisfied	Not satisfied	Yes	Suffered
SCF-MPEKS	Satisfied	Not satisfied	Yes	Suffered
tSCF-MPEKS	Satisfied	Satisfied	Yes	Suffered
Proposed Scheme	Satisfied	Satisfied	Yes	Not suffered

CT Ind, Trap Ind, MS and IKGA are the abbreviation of Ciphertext Indistinguishability, Trapdoor Indistinguishability, Multi-keywords Search and Inside Keyword Guessing Attack respectively. As seen from Table 1, the proposed scheme is more secure compared with the others. More specially, all of them excepting the proposed scheme and PAEKS are vulnerable to IKGA. Although PAEKS scheme prevents IKGA, it only aims for solving single keyword search problem instead of supporting multiple keywords search so that it may not be applied to the general public networks. To conclude, the proposed scheme has better performance and greater security than its counterparts.

Table 2 below compares the computation efficiency between the proposed scheme (PAEMKS) and its counterparts.

TABLE 2. Comparison of Computation Efficiency between PAEMKS and its counterparts

Scheme	PEKS	Trapdoor	Test
PEKS	2E + 2H + P	E + H	H + P
SCF-PEKS	2E + 2H + 2P	E + H	E + H + P
dPEKS	2E + 2H + P	3E + 2H	2E + 2H + P
PAEKS	3E + H	E + H + P	2P
MPEKS	E + (E + 2H + P) * Num	E + H	H + P
SCF-MPEKS	E + (E + 2H + P) * Num	E + H	E + H + P
tSCF-MPEKS	E + (E + 2H + P) * Num	3E + H + 2P	2E + H + 2P
Proposed Scheme	(2E + H + P) * Num	E + H + P	E

According to Table 2, the symbols E, H and P are the abbreviation of a modular exponentiation, a collision resistant hash function and a bilinear pairing respectively. Besides, Num denotes the number of keywords for searching. The PAEKS and proposed schemes which resist IKGA have the similar efficiency in Trapdoor algorithm. But the proposed scheme has better computation efficiency in Test algorithm than PAEKS scheme mainly because it only executes one modular exponentiation.

Table 3 summarised the communication efficiency between the proposed scheme (PAEMKS) and its counterparts.

TABLE 3. Comparison of Communication Efficiency between PAEMKS and its counterparts

Scheme	|PK|	|C|	$\|T_w\|$
PEKS	$\|G_1\|$	$\|G_1\| + n$	$\|G_1\|$
SCF-PEKS	$2\|G_1\|$	$\|G_1\| + n$	$\|G_1\|$
dPEKS	$2\|G_1\|$	$\|G_1\| + n$	$2\|G_1\|$
PAEKS	$\|G_1\|$	$2\|G_1\|$	$\|G_T\|$
MPEKS	$\|G_1\|$	$\|G_1\| + n * Num$	$\|G_1\|$
SCF-MPEKS	$\|G_1\|$	$\|G_1\| + n * Num$	$\|G_1\|$
tSCF-MPEKS	$2\|G_1\|$	$\|G_1\| + n * Num$	$\|G_1\| + 2\|G_T\|$
Proposed Scheme	$\|G_1\|$	$\|G_1\| + \|G_T\| * Num$	$\|G_T\|$

According to Table 3, the symbols of $|G_1|$ and $|G_T|$ devote the length of element in group G_1 and G_T. Besides, n and Num denote the length of security parameter and the number of keywords for searching. It is easy to see that the

proposed scheme has better communication efficiency than some of its counterparts. For instance, comparing with SCF-PEKS, dPEKS and tSCF-MPEKS schemes, the proposed scheme is efficient in $|PK|$. The proposed scheme also has greater efficiency in $|T_w|$ than tSCF-MPEKS scheme.

Table 4 below summarised the simulation platform of PAEMKS scheme. Note that the proposed scheme is programmed by JAVA and JPBC Library [25].

TABLE 4. Simulation platform for PAEMKS

OS	macOS Mojave 10.14.4
CPU	2.5 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Hard disk	512 GB
Programming language	JAVA

Figures 1 below compares KeyGen(S), KeyGen(R), PEKS, Trapdoor and Test generation algorithms between the tSCF-MPEKS scheme and the proposed scheme (PAEMKS) by 1000 times computer simulations. Every 100 times computer simulation is called one round. In KeyGen(S) generation algorithm, the proposed scheme is slightly efficient than the tSCF-MPEKS scheme. In KeyGen(R) generation algorithm, these two schemes are similar. However, the proposed scheme witnesses a high efficiency in PEKS, Trapdoor and Test generation algorithms comparing with the tSCF-MPEKS scheme.

6 CONCLUSION

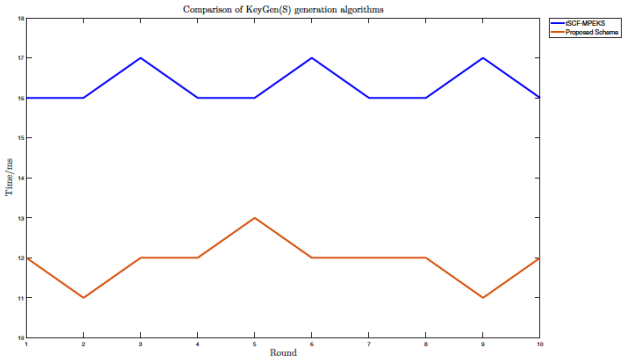
This paper firstly revisits Ma and Kazemian's tSCF-MPEKS scheme [17] and points out that the model is vulnerable to IKGA. Then, the paper defines the model of *Public Key Authenticated Encryption with Multi-keywords Search (PAEMKS)* and also presents a concrete construction of PAEMKS. This new scheme not only addresses Multiple Keywords Search issue but also incorporates with the advantages of User Authentication technique and Ciphertext Indistinguishability and Trapdoor Indistinguishability so that it is able to prevent IKGA. Therefore, the PAEMKS scheme is a secure, practical and cost-saving system. In addition, the efficiency and performance of the PAEMKS system is described by both theoretical analysis and computer simulations of 1000 times. Note that every 100 times is called one round. Furthermore, side channel analysis will be considered to PEKS in future works.

REFERENCES

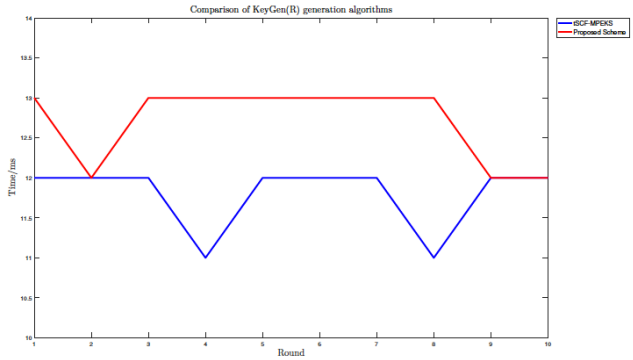
1. Jaber, A. and Mohamad Fadli Bin Zolkipli, 2013. Use of cryptography in cloud computing. 2013 IEEE International Conference on Control System, Computing and Engineering,.
2. Chun-Ta Li, Chin-Wen Lee and Jau-Ji Shen, 2015. A secure three-party authenticated key exchange protocol based on extended chaotic maps in cloud storage service. *2015 International Conference on Information Networking (ICOIN)*,.
3. Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G.: Public Key Encryption with Keyword Search, Advances in Cryptology, EUROCRYPT 2004, vol 3024, 506-522 (2004).
4. Byun J.W., Rhee H.S., Park HA., Lee D.H.: Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data, Secure Data Management 2006, vol 4165, 75-83 (2006).
5. Baek, J., Safavi-Naini, R. and Susilo, W.: Public Key Encryption with Keyword Search Revisited, Computational Science and Its Applications, ICCSA 2008, vol 5072, 1249-1259 (2008).
6. Yau, W., Heng, S. and Goi, B. (n.d.). Off-Line Keyword Guessing Attacks on Recent Public Key Encryption with Keyword Search Schemes. Lecture Notes in Computer Science, pp.100-105.
7. Jeong, I., Kwon, J., Hong, D. and Lee, D. (2009). Constructing PEKS schemes secure against keyword guessing attacks is possible?. Computer Communications, 32(2), pp.394-396.
8. Tang, Q. and Chen, L.: Public-Key Encryption with Registered Keyword Search, Public Key Infrastructures, Services and Applications, EuroPKI 2009, vol 6391, 163-178 (2010).
9. Rhee, H., Park, J., Susilo, W. and Lee, D.: Trapdoor security in a searchable public-key encryption scheme with a designated tester, Journal of Systems and Software, vol 83(5), 763-771 (2010).
10. Zhao, Y. J., Chen, X. F., Ma, H., Tang, Q., and Zhu, H.: A New Trapdoor indistinguishable Public Key Encryption with Keyword Search, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol 3, 72-81 (2013).
11. Yau, W., Phan, R., Heng, S. and Goi, B., 2013. Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester.

International Journal of Computer Mathematics, 90(12), pp.2581-2587.

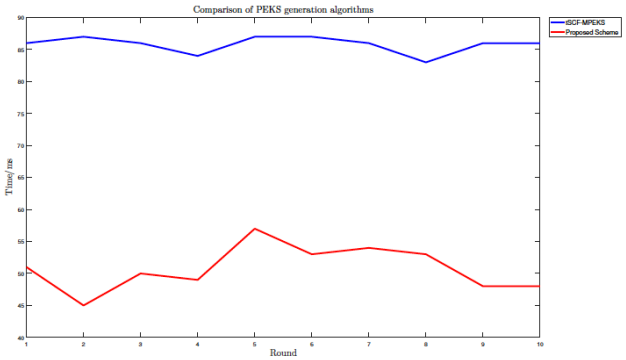
12. Chen YC (2015) SPEKS: secure server-designation public key encryption with keyword search against keyword guessing attacks. Comput J 58(4):922–933.
13. Noroozi, M., & Eslami, Z. (2019). Public-key encryption with keyword search: A generic construction secure against online and offline keyword guessing attacks. Journal of Ambient Intelligence and Humanized Computing, 1-12.
14. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K. and Lou, W.: Fuzzy Keyword Search over Encrypted Data in Cloud Computing, 2010 Proceedings IEEE INFOCOM, 257-266 (2010).
15. Xu, P., Jin, H., Wu, Q. and Wang, W.: Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack, IEEE Transactions on Computers, vol 62(11), 2266-2277 (2013).
16. Wang, T., Au, M. and Wu, W.: An Efficient Secure Channel Free Searchable Encryption Scheme with Multiple Keywords, Network and System Security, v9955, 251-265 (2016).
17. Ma, Y. and Kazemian, H. (2018). Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (Student Contributions). Proceedings of the 11th International Conference on Security of Information and Networks - SIN ’18.
18. Li, C.-T., Lee, C.-W., Shen, J.-J.: An extended chaotic maps-based keyword search scheme over encrypted data resist outside and inside keyword guessing attacks in cloud storage services. Nonlinear Dyn. 80(3), 1601–1611 (2015).
19. Noroozi, M., Eslami, Z., & Pakniat, N. (2018). Comments on a chaos-based public key encryption with keyword search scheme. Nonlinear Dynamics, 94(2), 1127-1132.
20. Huang, Q. and Li, H. (2017). An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Information Sciences, 403-404, pp.1-14.
21. Noroozi, M., & Eslami, Z. (2018). Public key authenticated encryption with keyword search: revisited. IET Information Security, 13(4), 336-342.
22. Zhang, X., Xu, C., Xie, R. and Jin, C. (2018). Designated Cloud Server Public Key Encryption with Keyword Search from Lattice in the Standard Model. Chinese Journal of Electronics, 27(2), pp.304-309.
23. Kazemian, H. and Ma, Y., 2020. Fuzzy Logic Application to Searchable Cryptography. Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference, pp.410-422.
24. Boneh, D. and Boyen, X.: Secure Identity Based Encryption Without Random Oracles, Advances in Cryptology, CRYPTO 2004, vol 3152, 443-459 (2004).
25. De Caro, A. and Iovino, V.: jPBC: Java pairing based cryptography, 2011 IEEE Symposium on Computers and Communications, 850-855 (2011).



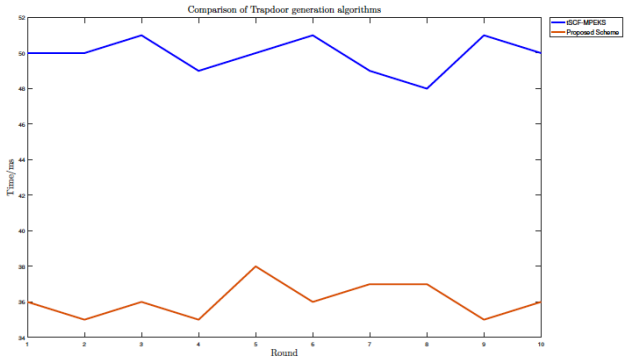
(a) Comparison of KeyGen(S) generation algorithms



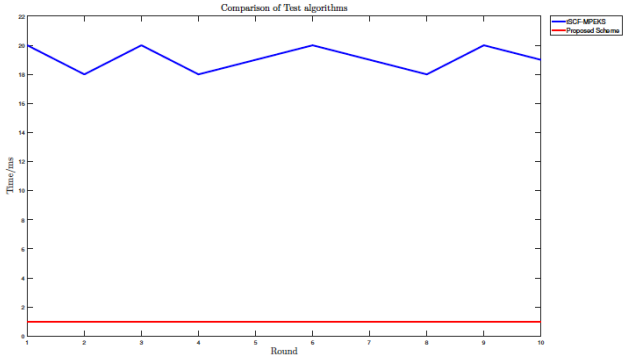
(b) Comparison of KeyGen(R) generation algorithms



(c) Comparison of PEKS generation algorithms



(d) Comparison of Trapdoor generation algorithms



(e) Comparison of Test generation algorithms

FIGURE 1. Comparison of Computer Simulation between the tSCF-MPEKS and the PAEMKS Schemes