K-Resilient Public Key Authenticated Encryption with Keyword Search

Koon-Ming Chan[a], Swee-Huay Heng[b], Syh-Yuan Tan[c] and Shing-Chiang Tan[d]
Faculty of Information Science and Technology, Multimedia University,
Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia

Keywords: PEKS, PAEKS, KGA.

Abstract: The public key encryption with keyword search (PEKS) scheme is a cryptographic primitive introduced to securely store and allow specific searches within encrypted data. Traditional encryption prioritises confidentiality but complicates search operations, requiring decryption before searches can be conducted. PEKS scheme addresses this limitation by enabling authorised users to search for specific keywords within encrypted data without compromising the underlying encryption. This facilitates efficient and secure data retrieval without the need to decrypt the entire dataset. However, PEKS is susceptible to the keyword guessing attack (KGA), exploiting the deterministic nature of the PEKS trapdoor so the adversary can correctly guess the keyword encrypted in a trapdoor. To enhance PEKS security to counter KGA, various schemes have been proposed. A notable one is public key authenticated encryption with keyword search (PAEKS). PAEKS combines authentication and encryption with keyword-based search functionalities, ensuring data source authentication, encrypted information security, and keyword-based searches. This approach offers a more robust and secure alternative to traditional PEKS. However, many existing PAEKS schemes rely on computationally exhaustive bilinear pairing. In this paper, we propose a PAEKS scheme based on *k*-resilient identity-based encryption without bilinear pairing. By using the provable security approach, we show that our proposed PAEKS scheme satisfies keyword privacy and trapdoor privacy. Furthermore, we also present a performance evaluation of our proposed PAEKS scheme with other related PAEKS schemes and show the efficiency of our proposed scheme.

1 INTRODUCTION

Public key encryption with keyword search (PEKS) is a cryptographic primitive proposed by Boneh et al. (Boneh et al., 2004) that is designed to allow secure storage of encrypted data while enabling specific searches within the encrypted content. Traditional encryption methods provide confidentiality but hinder search operations, as they necessitate decryption before conducting any searches. PEKS was proposed to address the limitations of traditional encryption methods, where conducting searches on encrypted data was either impractical or required decryption, leading to potential security risks. It provides a solution to the challenge of securely searching data without compromising the underlying encryption. Its primary objective is to permit authorised users to search for specific keywords within encrypted data without revealing the encrypted content. PEKS allows users to encrypt data with a public key and perform searches on the encrypted data for specific keywords by an authorised party who possesses the corresponding private key, without the need to decrypt the entire dataset.

The Keyword Guessing Attack (KGA) on the PEKS scheme is a vulnerability that exploits the deterministic nature of the PEKS trapdoor. This vulnerability was pointed out by Byun et al. (Byun et al., 2006), then followed by several attacks performed on a number of PEKS schemes showing vulnerability to KGA (Byun et al., 2006; Yau et al., 2008; Yau et al., 2013; Lu et al., 2019). In a KGA, adversaries repetitively guess and encrypt potential keywords, then by observing the resulting ciphertexts, the attackers can analyse and compare these encrypted keywords to glean patterns or similarities, enabling them to deduce potential matches between the ciphertext and the trapdoor. This attack compromises the security of PEKS by potentially revealing matches between guessed keywords and the trapdoor, potentially leading to unauthorised access to sensitive information.

Several related schemes have been proposed in the

[a] https://orcid.org/0000-0002-0367-4129
[b] https://orcid.org/0000-0003-3627-2131
[c] https://orcid.org/0000-0003-1182-1210
[d] https://orcid.org/0000-0002-1267-1894

Chan, K., Heng, S., Tan, S. and Tan, S.
K-Resilient Public Key Authenticated Encryption with Keyword Search.
DOI: 10.5220/0012716100003767

In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 381-388
ISBN: 978-989-758-709-2; ISSN: 2184-7711

literature to enhance the security of PEKS and counter the KGA attack. One significant scheme is a public key authenticated encryption with keyword search (PAEKS), introduced by Huang and Li (Huang and Li, 2017). It is designed to provide both authentication and encryption capabilities alongside keyword-based search functionalities. PAEKS allows users to authenticate the data source, encrypt the information, and perform searches based on keywords, ensuring both the confidentiality and integrity of the data. The advantage of PAEKS lies in its ability to combine authenticated encryption with keyword search, providing a more robust and secure approach compared to traditional PEKS. Up to now, many PAEKS schemes have been proposed (Lu and Li, 2022; Ma and Kazemian, 2021; Noroozi and Eslami, 2019; Qin et al., 2020) but most of the schemes are based on bilinear pairing which is known for its exhaustive computation.

In the effort to fill in the research gap of proposing a bilinear pairing-free PEKS scheme, in this paper, we mainly focus on the PAEKS scheme. We first propose the *k*-resilient PAEKS scheme without bilinear pairing based on the *k*-resilient PEKS scheme of (Yau et al., 2013) which was developed from Heng and Kurosawa's *k*-resilient identity-based encryption scheme (Heng and Kurosawa, 2004). We then follow the rigorous security analysis and show that our proposed PAEKS scheme satisfies keyword privacy and trapdoor privacy which also implies that it can resist KGA. Subsequently, we conduct a performance evaluation of our proposed PAEKS scheme with other related PAEKS schemes.

2 RELATED WORK

One effective method to prevent KGA is by resorting to a PAEKS scheme introduced earlier. A PAEKS scheme uses two sets of key pairs, the PAEKS scheme ciphertext requires a sender's private key and a receiver's public key to generate and vice versa for the trapdoor. It is for this reason that the PAEKS scheme was able to prevent KGA even from malicious servers. Li et al. (Li et al., 2019) proposed a designated-server identity-based PAEKS scheme that integrated the designated identity-based PEKS scheme with a PAEKS scheme. Noroozi and Eslami (Noroozi and Eslami, 2019) noted that Huang and Li's PAEKS scheme (Huang and Li, 2017) was insecure against KGA in a multi-user setting and they have also proposed an improvement to the scheme that fixed the issue. Qin et al. (Qin et al., 2020) also noted that Huang and Li's PAEKS scheme (Huang and Li, 2017) was vulnerable to multi-ciphertext attack. They proposed the first scheme in order to fix the vulnerability, followed by a second scheme that simplifies the management of the data sender's public key. Lu and Li (Lu and Li, 2022) proposed a designated PAEKS scheme without bilinear pairing. Instead of using only the sender's key pair and the receiver's key pair, it also requires an additional server's key pair. This limits their test algorithm to run on the designated server only. Huang et al. (Huang et al., 2023) proposed a more efficient PAEKS scheme using bilinear pairing. They incorporated ciphertext deduplication and inverted index to lower the storage cost and accelerate the searching process. Pu et al. (Pu et al., 2023) proposed a PAEKS scheme that was suitable for industrial IoT devices because it has one bilinear pairing operation in the testing algorithm. Bai et al. (Bai et al., 2024) also proposed a designated PAEKS scheme and their proposed scheme is more efficient than Lu and Li's designated PAEKS scheme (Lu and Li, 2022).

3 PRELIMINARIES

3.1 PAEKS Scheme

A PAEKS scheme consists of the following six algorithms:

- **Setup** (1^λ)**:** This algorithm takes security parameter 1^λ as the input to generate the common parameters cp that are required for the scheme.
- **SenderKeyGen** (cp)**:** This algorithm takes the common parameters cp to generate a set of public key and private key for the sender.
- **ReceiverKeyGen** (cp)**:** This algorithm takes the common parameters cp to generate a set of public key and private key for the receiver.
- **PAEKS** $(cp, w, sk_{sender}, pk_{receiver})$**:** This algorithm takes in the common parameters cp, a keyword w, sender's private key sk_{sender}, and receiver's public key $pk_{receiver}$ as input to generate a searchable ciphertext of the keyword C_w as an output.
- **Trapdoor** $(cp, w, pk_{sender}, sk_{receiver})$**:** This algorithm takes in the common parameters cp, a keyword w, sender's public key pk_{sender}, and receiver's private key $sk_{receiver}$ as input to generate a trapdoor of the keyword T_w as an output.
- **Test** (cp, C_w, T_w)**:** This algorithm is run by the server. It takes in a searchable ciphertext C_w and a trapdoor T_w to perform the matching operation.

If the searchable ciphertext matches with the trapdoor, it will output 1 else 0.

3.2 Security Models

3.2.1 Indistinguishability Under Chosen Keyword Attack (IND-CKA)

This model is defined by the following game between an adversary A and a challenger B:
Setup: The challenger B inputs the security parameter 1^{λ} to the Setup algorithm to compute the common parameters cp. Then, it runs the SenderKeyGen and ReceiverKeyGen algorithms to generate key pairs $(sk_{sender}, pk_{sender}, sk_{receiver}, pk_{receiver})$. In the end of this phase, B transfers $(cp, pk_{sender}, pk_{receiver})$ to the adversary A.

Phase 1: In this phase, the adversary A can adaptively query to the Ciphertext Oracle O_C and Trapdoor Oracle O_T.

- Ciphertext Oracle O_C: This oracle takes a keyword w as an input and outputs the ciphertext C_w by running the PAEKS algorithm. Finally, it returns the ciphertext C_w to A.
- Trapdoor Oracle O_T: This oracle takes a keyword w as an input and outputs the trapdoor T_w by running the Trapdoor algorithm. Finally, it returns the trapdoor T_w to A.

Challenge: The adversary selects two keywords $\{w_0, w_1\} \in W$ as the challenge keywords. The only restriction is that the challenge keywords should not have been queried for ciphertext and trapdoor previously. The challenger randomly chooses a bit $b \in \{0,1\}$ and returns the C_{w_b} to the adversary A.

Phase 2: In this phase, the adversary can still adaptively query to the O_C and O_T for any keywords except for the challenge keywords, i.e. $w \neq \{w_0, w_1\}$.

Guess: The adversary outputs a guess bit $b' \in \{0,1\}$. If $b' = b$, it wins the game.

We say that a PAEKS scheme possesses keyword privacy, that is, it achieves indistinguishability against a chosen keyword attack if any polynomial-time adversary A has a negligible advantage ε in the IND-CKA game, where

$$Adv_A^{IND-CKA} = |\Pr[b = b'] - \frac{1}{2}| \leq \varepsilon \qquad (1)$$

3.2.2 Indistinguishability Under Keyword Guessing Attack (IND-KGA)

This model is defined by the following game between an adversary A and a challenger B:

Setup: The challenger B inputs the security parameter 1^{λ} to the Setup algorithm to compute the common parameters cp. Then, it runs the SenderKeyGen and ReciverKeyGen algorithms to generate key pairs $(sk_{sender}, pk_{sender}, sk_{receiver}, pk_{receiver})$. In the end of this phase, B transfers $(cp, pk_{sender}, pk_{receiver})$ to the adversary A.

Phase 1: In this phase, the adversary A can adaptively query to the Ciphertext Oracle O_C and Trapdoor Oracle O_T.

Challenge: The adversary selects two keywords $\{w_0, w_1\} \in W$ as the challenge keywords. The only restriction is that the adversary cannot ask for the ciphertext of the challenge keywords C_{w_0} and C_{w_1}. The challenger randomly chooses a bit $b \in \{0,1\}$ and returns the T_{w_b} to the adversary A.

Phase 2: In this phase, the adversary can still adaptively query to the O_C and O_T for any keywords except for the challenge keywords, i.e. $w \neq \{w_0, w_1\}$.

Guess: The adversary outputs a guess bit $b' \in \{0,1\}$. If $b' = b$, it wins the game.

We say that a PAEKS scheme possesses trapdoor privacy, that is, it achieves indistinguishability against a keyword guessing attack if any polynomial-time adversary A has a negligible advantage ε in the IND-KGA game, where

$$Adv_A^{IND-KGA} = |\Pr[b = b'] - \frac{1}{2}| \leq \varepsilon \qquad (2)$$

4 PROPOSED KR-PAEKS SCHEME

Heng and Kurosawa (Heng and Kurosawa, 2004) first proposed the k-resilient identity-based encryption scheme in the standard model. It is well known in the literature that any anonymous identity-based encryption scheme can be transformed into a PEKS scheme following the transformation technique proposed by Abdalla et al. (Abdalla et al., 2005). Using the Heng and Kurosawa's scheme and Abdalla et al.'s transformation technique, Khader (Khader, 2006) proposed the k-resilient PEKS scheme. Khader's

scheme (Khader, 2006) was later improved by Yau et al. (Yau et al., 2012). They improved Khader's scheme (Khader, 2006) efficiency by removing and simplifying some unnecessary and complex steps in her overall scheme construction. However, the proposed k-resilient PEKS scheme is still vulnerable to KGA because it does not possess trapdoor privacy. We then propose the KR-PAEKS scheme possesses keyword privacy and trapdoor privacy based on Yau et al.'s (Yau et al., 2012) scheme.

Our proposed KR-PAEKS scheme consists of the following six algorithms:

KR-Setup: Choose a generator $g_1 \in \mathbb{G}$ and a random $a \in \mathbb{Z}_q^*$ to compute $g_2 = g_1^a$. Choose a random k. Set the common parameters $cp = \langle \mathbb{G}, g_1, g_2, k \rangle$.

KR-SenderKeyGen: Choose two random k-degree polynomials over $\mathbb{Z}_q$:

$$p_1(x) = \Sigma_{t=0}^{k} d_t x^t; \quad p_2(x) = \Sigma_{t=0}^{k} d_t^{'} x^t \tag{3}$$

Then, compute $D_t = g_1^{d_t} g_2^{d_t^{'}}$, for $0 \leq t \leq k$. Set the private key $sk_{sender} = \langle p_1, p_2 \rangle$ and the public key $pk_{sender} = \langle g_1, g_2, D_0, ..., D_k \rangle$.

KR-ReceiverKeyGen: Choose two random k-degree polynomials over $\mathbb{Z}_q$:

$$\hat{p}_1(x) = \Sigma_{t=0}^{k} \hat{d}_t x^t; \quad \hat{p}_2(x) = \Sigma_{t=0}^{k} \hat{d}_t^{'} x^t \tag{4}$$

Then, compute $\hat{D}_t = g_1^{\hat{d}_t} g_2^{\hat{d}_t^{'}}$, for $0 \leq t \leq k$. Set the private key $sk_{receiver} = \langle \hat{p}_1, \hat{p}_2 \rangle$ and the public key $pk_{receiver} = \langle g_1, g_2, \hat{D}_0, ..., \hat{D}_k \rangle$.

KR-PAEKS: Choose a random $r \in \mathbb{Z}_q^*$. For a chosen keyword w, compute

$$C_1 = (\Pi_{t=0}^{k} \hat{D}_t^{w^t})^{p_1(w)r} = (g_1^{\hat{p}_1(w)} \cdot g_2^{\hat{p}_2(w)})^{p_1(w)r}, \tag{5}$$

$$C_2 = (\Pi_{t=0}^{k} \hat{D}_t^{w^t})^{p_2(w)r} = (g_1^{\hat{p}_1(w)} \cdot g_2^{\hat{p}_2(w)})^{p_2(w)r}, \tag{6}$$

and

$$u = \frac{p_2(w)}{p_1(w)}. \tag{7}$$

Lastly, output the searchable ciphertext $C = \langle C_1, C_2, u \rangle$.

KR-Trapdoor: Choose a random $r^{'} \in \mathbb{Z}_q^*$. For a chosen keyword w, compute

$$T_1 = (\Pi_{t=0}^{k} D_t^{w^t})^{\hat{p}_1(w)r^{'}} = (g_1^{p_1(w)} \cdot g_2^{p_2(w)})^{\hat{p}_1(w)r^{'}}, \tag{8}$$

$$T_2 = (\Pi_{t=0}^{k} D_t^{w^t})^{\hat{p}_2(w)r^{'}} = (g_1^{p_1(w)} \cdot g_2^{p_2(w)})^{\hat{p}_2(w)r^{'}}, \tag{9}$$

and

$$\hat{u} = \frac{\hat{p}_2(w)}{\hat{p}_1(w)}. \tag{10}$$

Lastly, output the trapdoor $T_w = \langle T_1, T_2, \hat{u} \rangle$.

KR-Test: Upon receiving a trapdoor T_w, the server will test for a matching searchable ciphertext C by performing $(C_1)^u \cdot T_2 = (T_1)^{\hat{u}} \cdot C_2$. It will output "1" if the result matches, else output "0". The correctness of the equation is as follows:

$$(C_1)^u \cdot T_2 = (T_1)^{\hat{u}} \cdot C_2 \tag{11}$$

$$\left((g_1^{\hat{p}_1(w)} \cdot g_2^{\hat{p}_2(w)})^{p_1(w)r}\right)^{\frac{p_2(w)}{p_1(w)}} \cdot (g_1^{p_1(w)} \cdot g_2^{p_2(w)})^{\hat{p}_2(w)r^{'}} = \left((g_1^{p_1(w)} \cdot g_2^{p_2(w)})^{\hat{p}_1(w)r^{'}}\right)^{\frac{\hat{p}_2(w)}{\hat{p}_1(w)}} \cdot (g_1^{\hat{p}_1(w)} \cdot g_2^{\hat{p}_2(w)})^{p_2(w)r} \tag{12}$$

$$(g_1^{\hat{p}_1(w)} \cdot g_2^{\hat{p}_2(w)})^{p_2(w)r} \cdot (g_1^{p_1(w)} \cdot g_2^{p_2(w)})^{\hat{p}_2(w)r^{'}} = (g_1^{p_1(w)} \cdot g_2^{p_2(w)})^{\hat{p}_2(w)r^{'}} \cdot (g_1^{\hat{p}_1(w)} \cdot g_2^{\hat{p}_2(w)})^{p_2(w)r} \tag{13}$$

5 SECURITY ANALYSIS

This section presents the security proof of IND-CKA security and IND-KGA security of our scheme. It can guarantee that the keyword ciphertext and the trapdoor are indistinguishable repectively.

Theorem 1. *The proposed KR-PAEKS scheme achieves ciphertext indistinguishability under chosen keyword attack (IND-CKA) if the DDH assumption holds.*

Proof. Suppose there exists an adversary A who can (t, ε)-break the KR-PAEKS scheme in the IND-CKA security model, then we can construct a simulator B to (t', ε')-solve the DDH problem. Given as input a DDH instance (g, g^x, g^y, Z) over the cyclic group $(\mathbb{G}, g, p)$, B runs A and works as follows.

Setup: B sets the common parameters $cp = \langle \mathbb{G}, g_1 = g, g_2 = g^x, k \rangle$. Then, it runs the SenderKeyGen and ReceiverKeyGen algorithms to generate key pairs $(sk_{sender}, pk_{sender}, sk_{receiver}, pk_{receiver})$. In the end of this phase, B transfers $(cp, pk_{sender}, pk_{receiver})$ to the adversary A.

Phase 1: In this phase, the adversary A can adaptively query to the Ciphertext Oracle O_C and Trapdoor Oracle O_T of any keyword w.

Challenge: The adversary selects two keywords $\{w_0, w_1\} \in W$ as the challenge keywords. The only restriction is that the challenge keywords should not have been queried for ciphertext and trapdoor previously. B randomly chooses a bit $b \in \{0,1\}$ and returns the $C^*_{w_b}$ to the adversary A as

$$C^*_{w_b} = \{C^*_1, C^*_2, u\} = \Big\{ (g^{y \cdot \hat{p}_1(w_b) \cdot p_1(w_b)} \cdot Z^{\hat{p}_2(w_b) \cdot p_1(w_b)}), (g^{y \cdot \hat{p}_1(w_b) \cdot p_2(w_b)} \cdot Z^{\hat{p}_2(w_b) \cdot p_2(w_b)}), \frac{p_2(w_b)}{p_1(w_b)} \Big\}, \tag{14}$$

where g^y and Z are from the problem instance. The ciphertext correctness can be verified as follows:

$$\begin{aligned} C^*_{w_b} &= \Big\{ (g^{y \cdot \hat{p}_1(w_b) \cdot p_1(w_b)} \cdot Z^{\hat{p}_2(w_b) \cdot p_1(w_b)}), (g^{y \cdot \hat{p}_1(w_b) \cdot p_2(w_b)} \cdot Z^{\hat{p}_2(w_b) \cdot p_2(w_b)}), \frac{p_2(w_b)}{p_1(w_b)} \Big\} \\ &= \Big\{ (g^{r \cdot \hat{p}_1(w_b) \cdot p_1(w_b)} \cdot g^{x \cdot r \cdot \hat{p}_2(w_b) \cdot p_1(w_b)}), (g^{r \cdot \hat{p}_1(w_b) \cdot p_2(w_b)} \cdot g^{x \cdot r \cdot \hat{p}_2(w_b) \cdot p_2(w_b)}), \frac{p_2(w_b)}{p_1(w_b)} \Big\} \\ &= \Big\{ (g_1^{\hat{p}_1(w_b)} \cdot g_2^{\hat{p}_2(w_b)})^{p_1(w_b) r}, (g_1^{\hat{p}_1(w_b)} \cdot g_2^{\hat{p}_2(w_b)})^{p_2(w_b) r}, \frac{p_2(w_b)}{p_1(w_b)} \Big\}. \end{aligned} \tag{15}$$

where $r = y$ if $Z = g^{xy}$, and $r \in \mathbb{Z}^*_q$ is random if $Z \in \mathbb{G}$ is random. Therefore, $C^*_{w_b}$ is a valid challenge ciphertext whose encrypted keyword is w_b.

Phase 2: In this phase, the adversary A can still adaptively query to the O_C and O_T for any keywords except for the challenge keywords, i.e. $w \neq \{w_0, w_1\}$.

Guess: The adversary A outputs a guess bit $b' \in \{0,1\}$. The simulation outputs 1 if $b' = b$. Otherwise, it outputs 0.

This completes the simulation and the solution. The correctness is analysed as follows.

Probability of A Winning the Game.

- If $Z = g^{xy}$, the simulation is indistinguishable from the real attack, and thus the adversary A runs in time t and has a probability of $\varepsilon + \frac{1}{2}$ in guessing the encrypted keyword correctly.
- If Z is random, the challenge ciphertext is a one-time pad from the adversary's viewpoint. Therefore, the adversary A runs in time t and has a probability of $\frac{1}{2}$ in guessing the encrypted keyword correctly.

Advantage and Time Cost. When A wins the game, the simulator B (t', ε')-solves the DDH problem in time $t' = t$ and

$$\begin{aligned} \varepsilon' &= \Pr[A \text{ wins}] \\ &= |\Pr[b = b'] - \frac{1}{2}| \\ &= |\Pr[Z = g^{xy}] - \Pr[Z \text{ is random}]| = \varepsilon \end{aligned}$$

as required. This completes the proof of the Theorem1. □

Theorem 2. *The proposed KR-PAEKS scheme achieves trapdoor indistinguishability under keyword guessing attack (IND-KGA) if the DDH assumption holds.*

Proof. Suppose there exists an adversary A who can (t, ε)-break the KR-PAEKS scheme in the IND-KGA security model, then we can construct a simulator B to (t', ε)-solve the DDH problem. Given as input a DDH instance (g, g^x, g^y, Z) over the cyclic group $(\mathbb{G}, g, p)$, B runs A and works as follows.

Setup: B runs the Setup algorithm to compute the common parameters $cp = \langle G, g_1 = g, g_2 = g^x, k \rangle$. Then, it runs the SenderKeyGen and ReceiverKeyGen algorithms to generate key pairs $(sk_{sender}, pk_{sender}, sk_{receiver}, pk_{receiver})$. In the end of this phase, B transfers $(cp, pk_{sender}, pk_{receiver})$ to the adversary A.

Phase 1: In this phase, the adversary A can adaptively query to the Ciphertext Oracle O_C and Trapdoor Oracle O_T of any keyword w.

Challenge: The adversary selects two keywords $\{w_0, w_1\} \in W$ as the challenge keywords. The only restriction is that the adversary cannot ask for the ciphertext of the challenge keywords C_{w_0} and C_{w_1}. The challenger randomly chooses a bit $b \in \{0,1\}$ and returns the $T^*_{w_b}$ to the adversary A as

$$T^*_{w_b} = \langle T^*_1, T^*_2, \hat{u} \rangle = \Big\{ (g^{y \cdot \hat{p}_1(w_b) \cdot p_1(w_b)} \cdot Z^{\hat{p}_1(w_b) \cdot p_2(w_b)}), (g^{y \cdot \hat{p}_2(w_b) \cdot p_1(w_b)} \cdot Z^{\hat{p}_2(w_b) \cdot p_2(w_b)}), \frac{\hat{p}_2(w_b)}{\hat{p}_1(w_b)} \Big\} \tag{16}$$

where g^y and Z are from the problem instance. The

trapdoor correctness can be verified as follows:

$$
\begin{aligned}
T^*_{w_b} &= \Bigg\{ (g^{y\cdot\hat{p}_1(w_b)\cdot p_1(w_b)} \cdot Z^{\hat{p}_1(w_b)\cdot p_2(w_b)}), \\
&(g^{y\cdot\hat{p}_2(w_b)\cdot p_1(w_b)} \cdot Z^{\hat{p}_2(w_b)\cdot p_2(w_b)}), \frac{\hat{p}_2(w_b)}{\hat{p}_1(w_b)} \Bigg\} \\
&= \Bigg\{ (g^{r^{'}\cdot\hat{p}_1(w_b)\cdot p_1(w_b)} \cdot g^{x\cdot r^{'}\cdot\hat{p}_1(w_b)\cdot p_2(w_b)}), \\
&(g^{r^{'}\cdot\hat{p}_2(w_b)\cdot p_1(w_b)} \cdot g^{x\cdot r^{'}\cdot\hat{p}_2(w_b)\cdot p_2(w_b)}), \frac{\hat{p}_2(w_b)}{\hat{p}_1(w_b)} \Bigg\} \\
&= \Bigg\{ (g_1^{p_1(w_b)} \cdot g_2^{p_2(w_b)})^{\hat{p}_1(w_b)r^{'}}, (g_1^{p_1(w_b)} \\
&\cdot g_2^{p_2(w_b)})^{\hat{p}_2(w_b)r^{'}}, \frac{\hat{p}_2(w_b)}{\hat{p}_1(w_b)} \Bigg\}
\end{aligned}
\tag{17}
$$

where $r^{'} = y$ if $Z = g^{xy}$, and $r \in \mathbb{Z}_q^*$ is random if $Z \in \mathbb{G}$ is random. Therefore, $T^*_{w_b}$ is a valid challenge trapdoor for the keyword w_b.

Phase 2: In this phase, the adversary can still adaptively query to the O_C and O_T for any keywords except for the challenge keywords, i.e. $w \neq \{w_0, w_1\}$.

Guess: The adversary output a guess bit $b^{'} \in \{0,1\}$. The simulation outputs 1 if $b^{'} = b$. Otherwise, it outputs 0.

This completes the simulation and the solution. The correctness is analysed as follows.

Probability of A Winning the Game.

- If $Z = g^{xy}$, the simulation is indistinguishable from the real attack, and thus the adversary A has a probability of $\varepsilon + \frac{1}{2}$ in guessing the trapdoor correctly.
- If Z is random, the challenge ciphertext is a one-time pad from the adversary's viewpoint. Therefore, the adversary A has a probability of $\frac{1}{2}$ in guessing the trapdoor correctly.

Advantage and Time Cost. When A wins the game, the simulator B (t', ε')-solves the DDH problem in time $t' = t$ and

$$
\begin{aligned}
\varepsilon' &= \Pr[A \text{ wins}] \\
&= |\Pr[b = b^{'}] - \frac{1}{2}| \\
&= |\Pr[Z = g^{xy}] - \Pr[Z \text{ is random}]| = \varepsilon
\end{aligned}
$$

as required. This completes the proof of the Theorem 2. □

Table 1: Notation Used.

Notation	Description
M	Scalar multiplication operation in $\mathbb{G}$
M_t	Scalar multiplication operation in $\mathbb{G}_T$
m	Modular multiplication operation in $\mathbb{Z}_q$
A	Point addition operation in $\mathbb{G}$
a	Point addition operation in $\mathbb{Z}_q$
exp	Exponentiation operation in $\mathbb{Z}_q$
H	Hash operation
P	Bilinear pairing operation
inv	Modular inverse operation in $\mathbb{Z}_q$

6 PERFORMANCE EVALUATION

We compare our scheme with some related schemes in terms of computing efficiency. Table 1 defines the list of notation used in the performance evaluation. Table 2 shows the comparison analysis of our scheme with some related schemes in terms of the computational cost. In Table 2, we only consider the cost of Encryption algorithm, Trapdoor algorithm and the Test algorithm. We do not consider Setup algorithm and Key Generation algorithms in the comparison as they are executed once only.

To the best of our knowledge, our PAEKS scheme is the first construction without bilinear pairing. To perform a conservative comparison, although our scheme can be initialised with more efficient non-pairing curves such as ED25519 (which uses 256-bit modulus), we assume a pairing curve such as BLS-12 (which uses 392-bit modulus) is used. Subsequently, for clarity purposes, we standardise the complexity unit as the relative computation costs in equivalents of scalar multiplication in $\mathbb{G}_1$ under BLS-12 curve at 128-bit security (Tan and Groß, 2020). In precise, an exponentiation (M_t) in $\mathbb{G}_T$ is about the same as computing 6 scalar multiplication (M) in $\mathbb{G}_1$, and a pairing (P) is approximately $9M$. Table 3 shows the result after applying the conversion. The hashing and the operations in $\mathbb{Z}_q$ are typically not as significant compared to that in $\mathbb{G}$ but we keep it for completeness purposes.

From Table 3, it is evident that Pu et al.'s scheme (Pu et al., 2023) still has the advantage in the computation cost for the Encryption algorithm and the Trapdoor algorithm. However, a notable strength of our proposed scheme is in the Test algorithm, our scheme only requires $4M$ as compared to Pu et al.'s $9M$. Nevertheless, it is imperative to acknowledge that Pu et al.'s scheme holds a natural efficiency advantage over our proposed scheme due to their their

Table 2: Comparison Analysis of Computation Cost.

Scheme	Computation Cost			Security Model
	Encryption	Trapdoor	Test	
Huang and Li (2017) (Huang and Li, 2017)	$3M+H$	$M+H+P$	$2P$	Random Oracle
Noroozi and Eslami (2019) (Noroozi and Eslami, 2019)	$3M+H$	$M+H+P$	$2P$	Random Oracle
Qin et al. (2020) 1(Qin et al., 2020)	$3M+2H+P$	$2M+H$	$H+P$	Random Oracle
Qin et al. (2020) 2(Qin et al., 2020)	$2M+2H+2P$	$M+H+P$	$H+P$	Random Oracle
Huang et al. (2023)(Huang et al., 2023)	$3M+H$	$3M+H$	$2P$	Random Oracle
Pu et al. (2023)(Pu et al., 2023)	$3M+M_t$	M	P	Random Oracle
Proposed	$2((k+1)M)+2(k+1)exp+m+inv$	$2((k+1)M)+2(k+1)exp+m+inv$	$4M$	Standard

Table 3: Comparison Analysis in the Standardised Complexity Unit.

Scheme	Computation Cost			Security Model
	Encryption	Trapdoor	Test	
Huang and Li (2017) (Huang and Li, 2017)	$3M+H$	$10M+H$	$18M$	Random Oracle
Noroozi and Eslami (2019) (Noroozi and Eslami, 2019)	$3M+H$	$10M+H$	$18M$	Random Oracle
Qin et al. (2020) 1(Qin et al., 2020)	$12M+2H$	$2M+H$	$9M+H$	Random Oracle
Qin et al. (2020) 2(Qin et al., 2020)	$20M+2H$	$10M+H$	$9M+H$	Random Oracle
Huang et al. (2023)(Huang et al., 2023)	$3M+H$	$3M+H$	$18M$	Random Oracle
Pu et al. (2023)(Pu et al., 2023)	$9M$	M	$9M$	Random Oracle
Proposed	$2((k+1)M)+2(k+1)exp+m+inv$	$2((k+1)M)+2(k+1)exp+m+inv$	$4M$	Standard

scheme is provably secure in the random oracle model whereas our proposed scheme is provably secure in the standard model. Note that, we can further reduce the computation cost of our scheme by applying the multiple exponentiation algorithm (Menezes et al., 1996) for the Encryption algorithm and Trapdoor algorithm which can be reduced to $1.5((k+1)M) + 1.5(k+1)exp + m + inv$ for both Encryption algorithm and Trapdoor algorithm.

ACKNOWLEDGEMENT

This work was supported by the Telekom Malaysia Research & Development Grant (RDTC/221045).

REFERENCES

Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., and Shi, H. (2005). Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Advances in Cryptology–CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings 25*, pages 205–222. Springer.

Bai, L., Yong, L., Chen, Z., and Shao, J. (2024). Pairing-free public-key authenticated encryption with keyword search. *Computer Standards & Interfaces*, 88:103793.

Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 506–522. Springer.

Byun, J. W., Rhee, H. S., Park, H.-A., and Lee, D. H. (2006). Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Workshop on secure data management*, pages 75–83. Springer.

Heng, S.-H. and Kurosawa, K. (2004). k-resilient identity-based encryption in the standard model. In *Topics in Cryptology–CT-RSA 2004: The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, pages 67–80. Springer.

Huang, Q., Huang, P., Li, H., Huang, J., and Lin, H. (2023). A more efficient public-key authenticated encryption scheme with keyword search. *Journal of Systems Architecture*, 137:102839.

Huang, Q. and Li, H. (2017). An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14.

Khader, D. (2006). Public key encryption with keyword search based on k-resilient ibe. In *International Conference on Computational Science and Its Applications*, pages 298–308. Springer.

Li, H., Huang, Q., Shen, J., Yang, G., and Susilo, W. (2019). Designated-server identity-based authenticated encryption with keyword search for encrypted emails. *Information Sciences*, 481:330–343.

Lu, Y. and Li, J. (2022). Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices. *IEEE Transactions on Mobile Computing*, 21(12):4397–4409.

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

Ma, Y. and Kazemian, H. (2021). Public key authenticated encryption with multiple keywords search using mamdani system. *Evolving Systems*, 12(3):687–699.

Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (1996). Handbook of applied cryptography. *CRC, Boca Raton*, 17.

Noroozi, M. and Eslami, Z. (2019). Public key authenticated encryption with keyword search: revisited. *IET Information Security*, 13(4):336–342.

Pu, L., Lin, C., Chen, B., and He, D. (2023). User-friendly public-key authenticated encryption with keyword search for industrial internet of things. *IEEE Internet of Things Journal*.

Qin, B., Chen, Y., Huang, Q., Liu, X., and Zheng, D. (2020). Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Information Sciences*, 516:515–528.

Tan, S.-Y. and Groß, T. (2020). Monipoly—an expressive q-sdh-based anonymous attribute-based credential system. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 498–526. Springer.

Yau, W.-C., Heng, S.-H., and Goi, B.-M. (2008). Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In *Autonomic and Trusted Computing: 5th International Conference, ATC 2008, Oslo, Norway, June 23-25, 2008 Proceedings 5*, pages 100–105. Springer.

Yau, W.-C., Heng, S.-H., Tan, S.-Y., Goi, B.-M., and Phan, R. C.-W. (2012). Efficient encryption with keyword search in mobile networks. *Security and Communication Networks*, 5(12):1412–1422.

Yau, W.-C., Phan, R. C.-W., Heng, S.-H., and Goi, B.-M. (2013). Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. *International Journal of Computer Mathematics*, 90(12):2581–2587.