Journal of Intelligent Computing and Networking
https://www.ffspub.com/index.php/jicn/index
ISSN 3079-9228 (print)
E-mail: jicn.office@ffspub.com

Article

Pairing-Free Public-Key Authenticated Encryption with Keyword Search Resistant to Frequency Analysis for IoT

Kunlong Jin[1,2,3], Kaiqi Liu[1,2,3,†], Lujia Shi[1,2,3], Shuai Wang[1,2,3], Jiaen Zhou[4]
[1]*State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guizhou 550025, China*
[2]*Guizhou Provincial Key Laboratory of Cryptography and Blockchain Technology, Guizhou University, Guizhou 550025, China*
[3]*Institute of Cryptography and Data Security, Guizhou University, Guizhou 550025, China*
[4]*School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK*
[†]*E-mail: kqiliu@163.com*

Received: September 30, 2025 / Revised: November 12, 2025 / Accepted: November 17, 2025 / Published online: November 27, 2025

Abstract: To enhance the searchability of encrypted cloud data while preserving user privacy, public-key encryption with keyword search (PEKS) has been regarded as a promising approach. However, existing schemes still incur substantial computational overhead in resource-constrained IoT environments. Moreover, IoT applications frequently reuse certain keywords during operations such as data labeling and status reporting, making them more susceptible to frequency-analysis attacks. To address this, this paper analyzes the limitations of existing pairing-free public-key authenticated encryption with keyword search (PAEKS) schemes in resisting frequency analysis and proposes a pairing-free PAEKS construction based on elliptic-curve scalar multiplication. A probabilistic trapdoor is further introduced to weaken the linkability between keywords and their occurrence frequencies. The proposed scheme effectively mitigates frequency-analysis attacks and, by eliminating costly bilinear pairings, significantly reduces computational burden. Experimental results show that, compared with conventional schemes, the proposed approach achieves lower runtime in ciphertext and trapdoor generation while providing stronger protection against frequency analysis, thereby attaining a more favorable security–efficiency trade-off suitable for IoT deployments with constrained computation and bandwidth.

Keywords: Data security; Internet of Things; searchable public key encryption; frequency analysis

1 Introduction

With the widespread adoption of the Internet of Things (IoT) and 5G technologies, the number of edge devices has increased dramatically, leading to exponential growth of edge data [1]. To efficiently manage and utilize such data, it is commonly outsourced to cloud servers. Consequently, the emergence of cloud-assisted IoT (CIoT) has significantly transformed data access and management across various domains, including industrial automation, smart grids, healthcare, and vehicular networks. In these applications, data collected periodically by IoT devices is aggregated at gateways and then uploaded to the cloud for centralized storage, enabling real-time monitoring and analysis. Sensor data stored on cloud platforms can be accessed through user authorization, thereby supporting decision-making and system optimization. However, while CIoT enhances data accessibility and processing efficiency, it also introduces new security and privacy concerns. Although encryption protects sensitive information, it inevitably increases the complexity of data utilization in cloud environments, especially in terms of data retrieval.

1.1 Related Work

Searchable Encryption (SE) [2] is a promising cryptographic technique that enables secure search over encrypted data without decryption, while preserving the confidentiality of both search keywords and associated data files. Among existing SE schemes, Symmetric Searchable Encryption (SSE) [3] has been widely adopted for processing large-scale datasets due to its simple structure and high retrieval efficiency. However, most SSE schemes are designed primarily for general search applications. Although they provide advantages in terms of algorithmic simplicity and efficiency, they suffer

[†] Corresponding author: Kaiqi Liu
[*]Academic Editor: Xiuli Bi

from security limitations in key distribution, making them unsuitable for dynamic IoT environments or public cloud storage settings. In particular, static SSE schemes [4] fail to support data updates, rendering them impractical in scenarios where data is continuously generated and frequently uploaded.

To address these issues, PEKS was introduced [5]. By leveraging the receiver's public/private key pair to generate ciphertexts and trapdoors, PEKS eliminates the challenges of key sharing and distribution, thereby exhibiting strong potential in cloud storage environments [6, 7]. However, PEKS remains vulnerable to Keyword Guessing Attacks (KGA) [8], since in practice the keyword space is often small and the entropy of search keywords is relatively low, enabling adversaries to infer keyword information from trapdoors. KGA can be classified into external KGA, launched by outside attackers, and internal KGA, typically initiated by the server. To defend against KGA, Hu et al.[9] proposed a secure PEKS scheme against external attacks under a designated-server model. Subsequently, PAEKS [10, 11] was introduced, which applies signature encryption to keywords, combining encryption with digital signatures to provide effective protection against KGA. Later, PAEKS was extended into certificateless cryptographic settings, where researchers proposed various certificateless PAEKS schemes [12, 13], further enhancing both security and flexibility [14].

Nevertheless, deploying PAEKS schemes in CIoT environments still encounters two major challenges: security and computational overhead. On the one hand, PAEKS is vulnerable to frequency analysis attacks [15]. Although several countermeasures have been proposed to mitigate KGA [16, 17], adversaries can still compromise data privacy through frequency analysis attacks. In CIoT applications, specific keywords are frequently used in operations such as data tagging and status reporting, while differences in device types and industrial processes lead to uneven keyword distributions. By leveraging publicly available technical documents, industry reports, and historical datasets, adversaries can estimate keyword frequency distributions. Through frequency comparison of ciphertexts or trapdoors, attackers can infer plaintext keywords; even coarse-grained frequency knowledge significantly improves the success rate of frequency analysis attacks [18], thereby posing a severe threat to data privacy in CIoT scenarios. On the other hand, IoT devices in CIoT are typically resource-constrained, making lightweight cryptographic schemes essential for practical deployment [19]. However, existing PAEKS schemes heavily rely on bilinear pairings [20, 21], which are computationally intensive and costly. Compared to other common cryptographic operations, such as elliptic curve scalar multiplication, bilinear pairings require substantially more computation time, resulting in significant overhead. Since bilinear pairings fail to meet the lightweight and efficiency requirements of CIoT environments, they are unsuitable for deployment on resource-limited IoT devices.

1.2 Our Contributions

To address the dual challenges of security and computational overhead in applying PAEKS to CIoT environments, we propose a pairing-free PAEKS scheme based on elliptic curve scalar multiplication. The proposed scheme not only effectively resists frequency analysis attacks but also achieves a balance between security and efficiency in resource-constrained scenarios, thereby meeting the two critical requirements of PAEKS in CIoT. The main contributions of this paper are summarized as follows:

- We conduct an in-depth cryptanalysis of existing lightweight searchable encryption schemes, demonstrating how frequency analysis attacks can extract target keyword information embedded in trapdoors, thereby exposing the potential security risks of employing these lightweight schemes in CIoT environments.
- To simultaneously ensure lightweight design and strong security, we adopt a probabilistic trapdoor generation algorithm to construct a concrete PAEKS scheme. This approach mitigates the vulnerability of probabilistic and deterministic trapdoor generation methods to frequency analysis attacks in most existing schemes. Furthermore, the proposed scheme eliminates the need for computationally expensive bilinear pairings, thereby improving efficiency and practicality and achieving a lightweight yet effective solution.
- We perform a performance comparison with various state-of-the-art PAEKS constructions. The evaluation results show that our scheme maintains a high level of security while delivering superior performance.

1.3 Outline

The remainder of this paper is organized as follows: Section 2 provides an overview of the system and security models for our attack scenario. Section 3 details our proposed PAEKS scheme. Section 4 presents a security analysis of the scheme. Section 5 conducts comparative experiments, and finally, Section 6 concludes the paper.

2 Problem Formulation

This section first presents the system model of PAEKS and its corresponding security model. Then, a cryptanalysis is conducted to examine the security concerns of applying existing pairing-free schemes in CIoT environments.

2.1 System Model

The proposed PAEKS system model consists of three distinct entities, as illustrated in Figure 1: the cloud server (CS), the data owner (DO), and the data receiver (DR). It operates in six steps: keyword extraction, keyword encryption, ciphertext transmission, trapdoor generation, trapdoor transmission, and result retrieval.

1) Data Owner: The data owner first extracts keywords from the data files and encrypts them using the receiver's public key to generate keyword ciphertexts. The keyword ciphertexts are then uploaded to the cloud server together with the encrypted data files.

2) Data Receiver: The data receiver generates trapdoors for the target keywords and submits them to the cloud server as search queries. Finally, the receiver decrypts the encrypted

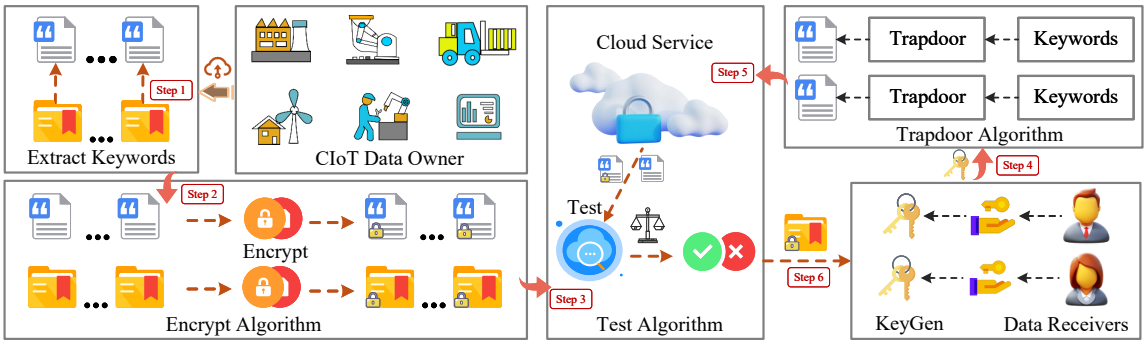

Figure 1: The PAEKS framework.

data files returned by the cloud server to obtain the desired information.

3) Cloud Server: The cloud server stores the encrypted data files and keyword ciphertexts uploaded by the data owner. Upon receiving a search request, it matches the keyword ciphertexts against the submitted trapdoors and returns the corresponding results.

Definition 1. A PAEKS scheme with a cloud server consists of five polynomial-time algorithms.

$\mathrm{Setup}(\lambda) \to \mathrm{params}$*:* Given a security parameter, output the system public parameters λ.

$\mathrm{KeyGen}(\mathrm{params}) \to (\mathrm{PK_U}, \mathrm{SK_U})$*:* Given a public parameter *params* as input, generate a pair of user keys $(\mathrm{PK_{ID_U}}, \mathrm{SK_{ID_U}})$.

$\mathrm{PAEKS}(w, \mathrm{SK_O}, \mathrm{PK_R}, \mathrm{PK_O}, \mathrm{PK_S}) \to C_w$*:* Input the public parameter params, a keyword w, the data owner's private key $\mathrm{SK_O}$, the data recipient's public key $\mathrm{PK_R}$, and the cloud server's public key $\mathrm{PK_S}$. Return an encrypted keyword C_w.

$\mathrm{Trapdoor}(w, \mathrm{SK_R}, \mathrm{PK_O}, \mathrm{PK_R}, \mathrm{PK_S}) \to T_w$*:* Input the public parameter params, a keyword w, the data receiver's private key $\mathrm{SK_R}$, the data owner's public key $\mathrm{PK_O}$, and the cloud server's public key $\mathrm{PK_S}$. Return a keyword trapdoor T_w.

$\mathrm{Test}(C_w, T_w) \to \mathrm{result}$*:* Given ciphertext keyword C_w and trapdoor T_w as input. If C_w and T_w share the same keyword, return the ciphertext of the data file; otherwise, return 0.

2.2 Security Model

In this section, we address Multi-Ciphertext Indistinguishability (MCI) and Multi-Trapdoor Indistinguishability (MTI). The security model for this scheme is defined through the following secure game between two challengers and adversary $\mathscr{A}$.

Game 1: Indistinguishability under multiple ciphertexts

Setup. Given a security parameter λ, challenger $\mathscr{C}$ executes the Setup algorithm and KeyGen algorithm, then returns the parameters and public key $(\mathrm{params}, \mathrm{PK_O}, \mathrm{PK_R}, \mathrm{PK_S})$ to adversary $\mathscr{A}$.

Phase 1. At this stage, adversary $\mathscr{A}$ permits multiple adaptive queries to challenger $\mathscr{C}$ using the ciphertext oracle and trapdoor oracle.

1) Ciphertext Query. Adversary $\mathscr{A}$ randomly selects a keyword w and a recipient's public key PK to send to the challenger. In response, $\mathscr{C}$ executes $C_w \leftarrow \mathrm{PAEKS}(w, \mathrm{SK_O}, \mathrm{PK'_R}, \mathrm{PK_O}, \mathrm{PK_S})$ and returns it to adversary $\mathscr{A}$.

2) Trapdoor query. $\mathscr{A}$ randomly selects a keyword w' and a data owner's public key $\mathrm{PK'_O}$ to send to the challenger. In response, $\mathscr{C}$ executes $T_{w'} \leftarrow \mathrm{Trapdoor}(w, \mathrm{SK_R}, \mathrm{PK'_O}, \mathrm{PK_R}, \mathrm{PK_S})$ and returns the result to adversary $\mathscr{A}$.

Challenge: Adversary $\mathscr{A}$ selects two keyword pairs $w_0 = (w^*_{0,i})_{i\in[1,n]}$ and $w_1 = (w^*_{1,i})_{i\in[1,n]}$, where keywords such as $(\mathrm{PK_O}, w^*_{0,i})$ and $(\mathrm{PK_O}, w^*_{1,i})$ have not been queried for ciphertext. Challenger $\mathscr{C}$ randomly selects $\delta \in \{0,1\}$ to generate challenge ciphertext $C_{w^*_{\delta,i}}$ and returns it to adversary $\mathscr{A}$.

Phase 2. Similar to Phase 1, Adversary $\mathscr{A}$ may continue to perform polynomial-time adaptive queries. However, the restriction is that Adversary $\mathscr{A}$ may not query any challenge keywords from the challenge phase.

Guess. Adversary $\mathscr{A}$ outputs his prediction $\delta' \in \{0,1\}$. The advantage for adversary $\mathscr{A}$ to win this game is defined as

$$Adv^{\mathrm{MCI}}_{\mathscr{A}}(\lambda) = |\Pr[\delta = \delta'] - \frac{1}{2}| \tag{1}$$

Game 2: Indistinguishability under multiple trapdoors

Setup. This stage corresponds to the initialization phase of the aforementioned security game with indistinguishability among multiple ciphers.

Phase 1. At this stage, adversary $\mathscr{A}$ permits multiple adaptive queries to challenger $\mathscr{C}$ using the ciphertext oracle and trapdoor oracle.

1) Ciphertext Query. $\mathscr{A}$ randomly selects a keyword w and a recipient's public key $\mathrm{PK'_R}$, sending them to challenger $\mathscr{C}$. In response, $\mathscr{C}$ executes $C_w \leftarrow \mathrm{PAEKS}(w, \mathrm{SK_O}, \mathrm{PK'_R}, \mathrm{PK_O}, \mathrm{PK_S})$ and returns the result to adversary $\mathscr{A}$.

2) Trapdoor query. $\mathscr{A}$ randomly selects a keyword w' and a data owner's public key $\mathrm{PK'_O}$ to send to the challenger. In response, $\mathscr{C}$ executes $T_{w'} \leftarrow \mathrm{Trapdoor}(w, \mathrm{SK_R}, \mathrm{PK'_O}, \mathrm{PK_R}, \mathrm{PK_S})$ and returns the result to adversary $\mathscr{A}$.

Challenge: Adversary $\mathscr{A}$ selects two keyword pairs $w_0 = (w^*_{0,i})_{i\in[1,n]}$ and $w_1 = (w^*_{1,i})_{i\in[1,n]}$, where keywords

such as $(\mathrm{PK_R}, w^*_{0,i})$ and $(\mathrm{PK_R}, w^*_{1,i})$ have not been queried in trapdoor queries. Challenger $\mathscr{C}$ randomly selects $\delta \in \{0,1\}$ to generate challenge trapdoor $T_{w^*_{\delta,i}}$ and returns it to adversary $\mathscr{A}$.

Phase 2. Similar to Phase 1, Adversary $\mathscr{A}$ may continue to perform polynomial-time adaptive queries. However, the restriction is that Adversary $\mathscr{A}$ may not query any challenge keywords from the challenge phase.

Guess. Adversary $\mathscr{A}$ outputs his prediction $\delta' \in \{0,1\}$. The advantage for adversary $\mathscr{A}$ to win this game is defined as

$$Adv^{\mathrm{MTI}}_{\mathscr{A}}(\lambda) = |\Pr[\delta = \delta'] - \frac{1}{2}| \tag{2}$$

2.3 Cryptanalysis of Existing Unpaired Schemes

In this section, we present a specific method to examine whether any two trapdoors embed the same key. Subsequently, attackers can perform frequency analysis on trapdoors classifications.

2.3.1 Cryptanalysis of the Lu–Li Scheme

In this subsection, we analyze the scheme proposed by Lu and Li [16]. f_1, f_2 are hash functions. The data owner and data recipient possess public keys $(\mathrm{PK_{A1}}, \mathrm{PK_{A2}})$ and $(\mathrm{PK_{B1}}, \mathrm{PK_{B2}})$, respectively, while the data recipient holds the private key $(\mathrm{SK_{B1}}, \mathrm{SK_{B2}})$. Therefore, the ST_W keyword trapdoor can be calculated as:

$$\mathrm{ST_w} = f_2(w, \lambda_1, \lambda_2)\mathrm{SK_{B1}} \tag{3}$$

Among these, λ_1 and λ_2 are two secret values shared exclusively between the data owner and the recipient to enhance binding and resistance to attacks. They can be calculated using the following formula:

$$\begin{aligned} \lambda_1 &= f_1(\mathrm{PK_{A1}}, \mathrm{PK_{B2}}, \mathrm{SK_{B1}PK_{B1}}) \\ \lambda_2 &= f_2(\mathrm{PK_{A2}}, \mathrm{PK_{B2}}, \mathrm{SK_{B2}PK_{B2}}) \end{aligned} \tag{4}$$

Clearly, since the trapdoor generation algorithm in this scheme is deterministic, identical keywords will produce identical trapdoors. Consequently, this scheme is highly susceptible to frequency analysis.

2.3.2 Cryptanalysis of the Liu–Sun–Dong Scheme for IoMT

We next analyze the scheme proposed by Liu, Sun, and Dong for IoMT [14]. H_1, H_2, H_3 are hash functions. The system's master public key and private key are $P_{pub} = sP$ and s, respectively. The identity identifiers for the data owner and data recipient are ID and IDo, with corresponding public keys (Y_{ID_O}, Q_{ID_O}) and (Y_{ID_R}, Q_{ID_R}). The data recipient's private key is (x_{ID_R}, d_{ID_R}), where $u_{ID_x} = H_1(ID_x, Q_{ID_x})$ denotes the binding coefficient between these two identities. Let the trapsets for keywords w_1 and w_2 be $T_{1,2}, T_{1,3})$ and $T_{w_2} = (T_{2,1}$, respectively. Select one-time random numbers $r, r' \in \mathbb{Z}^*_q$. Therefore, the aforementioned vector can be computed as:

$$\begin{aligned} T_{1,1} &= rP \\ T_{1,2} &= (r + \beta_{ID_R} H_2(w_1))(x_{ID_R} + d_{ID_R})^{-1} + \beta_{ID_R} \\ T_{1,3} &= \beta_{ID_R}(Y_{ID_O} + Q_{ID_O} + u_{ID_O} P_{pub}) \\ T_{2,1} &= r'P \\ T_{2,2} &= (r' + \beta_{ID_R} H_2(w_2))(x_{ID_R} + d_{ID_R})^{-1} + \beta_{ID_R} \\ T_{2,3} &= \beta_{ID_R}(Y_{ID_O} + Q_{ID_O} + u_{ID_O} P_{pub}) \end{aligned} \tag{5}$$

Here, β_{ID_R} represents the binding factor that associates the sender/receiver identity with both parties' public and private keys, denoted as

$$\beta_{ID_R} = H_3(ID_O, ID_R, x_{ID_R}, Y_{ID_O}) \in \mathbb{Z}^*_q \tag{6}$$

We can then derive:

$$\begin{aligned} &(T_{2,2} - T_{1,2})(Y_{ID_R} + Q_{ID_R} + u_{ID_R} P_{pub}) \\ &= (r + \beta_{ID_R} H_2(w_1) - (r' + \beta_{ID_R} H_2(w_2)))P \end{aligned} \tag{7}$$

Therefore, an attacker need only determine $(T_{2,2} - T_{1,2})(Y_{ID_R} + Q_{ID_R} + u_{ID_R} P_{pub}) = (T_{2,1} - T_{1,1})$ to verify whether $w_1 = w_2$ holds true.

2.3.3 Cryptanalysis of the Liu–Dong–Kumari–Kar Scheme for IIoT

Finally, we analyze the scheme proposed by Liu, Dong, Kumari, and Kar for IIoT [17]. H_1, H_2, H_3 are hash functions, with the system's master public key and private key being $P_{pub} = sP$ and s respectively. The identity identifiers for the data owner and data recipient are ID_O and ID_O, with public keys (Y_{ID_O}, Q_{ID_O}) and (Y_{ID_R}, Q_{ID_R}) respectively, and the data recipient's private key being (x_{ID_R}, d_{ID_R}). $u_{ID_x} = H_1(ID_x, Q_{ID_x})$ denotes the binding coefficient between these two identities. Select a one-time random number $r, r' \in \mathbb{Z}^*_q$. Let the trapsets for keywords w_1 and w_2 be $T_{w_1} = (t_{1,1}, t_{1,2}, t_{1,3})$ and $T_{w_2} = (t_{2,1}, t_{2,2}, t_{2,3})$, respectively. Therefore, the above vector can be computed as:

$$\begin{aligned} t_{1,1} &= r \in \mathbb{Z}^*_q \\ t_{1,2} &= \beta_{Ro}(Y_{ID_O} + Q_{ID_O} + u_{ID_O} P_{pub}) \\ 2t_{1,3} &= (1 + \beta_{Ro}(x_{ID_R} + d_{ID_R}))^{-1}(H_2(w_1) - t_{1,1}(x_{ID_R} + d_{ID_R}) \\ t_{2,1} &= r' \in \mathbb{Z}^*_q \\ t_{2,2} &= \beta_{Ro}(Y_{ID_O} + Q_{ID_O} + u_{ID_O} P_{pub}) \\ 2t_{2,3} &= (1 + \beta_{Ro}(x_{ID_R} + d_{ID_R}))^{-1}(H_2(w_2) - t_{2,1}(x_{ID_R} + d_{ID_R})) \end{aligned} \tag{8}$$

In this context, β_{RO} is a binding coefficient derived from a hash function that strongly binds the identities and key materials of the data owner DO and the data receiver DR.

For T_{w_1} and T_{w_2}, we can obtain

$$\begin{aligned} &(2t_{1,3})(P + Y_{ID_R} + Q_{ID_R} + u_{ID_R} P_{pub}) + t_{1,1}(Y_{ID_R} \\ &+ Q_{ID_R} + u_{ID_R} P_{pub}) \\ &= H_2(w_1)P \\ &(2t_{2,3})(P + Y_{ID_R} + Q_{ID_R} + u_{ID_R} P_{pub}) + t_{2,1}(Y_{ID_R} \\ &+ Q_{ID_R} + u_{ID_R} P_{pub}) \\ &= H_2(w_2)P \end{aligned} \tag{9}$$

Therefore, the adversary only needs to determine whether $H_2(w_1)P = H_2(w_2)P$ holds true to verify $w_1 = w_2$.

3 The Proposed Paeks Scheme

In this section, we present a secure PAEKS scheme resistant to frequency analysis attacks. The specific construction of the scheme is described as follows.

3.1 Construction

- Setup(λ). Given a security parameter λ as input, this algorithm generates an elliptic curve group $\mathbb{G}$ of order q based on the security parameter, where P is a generator of group $\mathbb{G}$. Then, a hash function $H : \mathbb{G} \times \mathbb{G} \times \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_q^*$ is selected. Subsequently, the system parameter params $= (\lambda, G, q, P, H)$ is made public.
- KeyGen(params). For the data owner, given params as input, the data owner selects a random number $x \in \mathbb{Z}_q^*$, calculates $\mathrm{PK_O} = xP$, and sets $\mathrm{SK_O} = x$.

  For the data recipient, given params as input, the data recipient selects a random number $y \in Z_q^*$, calculates $\mathrm{PK_R} = yP$, and sets $\mathrm{SK_R} = y$.

  For the cloud server, given params as input, the cloud server selects a random number $s \in Z_q^*$, calculates $\mathrm{PK_S} = sP$, and sets $\mathrm{SK_S} = s$.
- PAEKS($\boldsymbol{w}, \mathrm{SK_O}, \mathrm{PK_R}, \mathrm{PK_O}, \mathrm{PK_S}$). First, the data owner extracts keywords from the data file to obtain the keyword set $W = \{w_1, \cdots, w_n\}$ of the encrypted data file.

  Given params, the data recipient's public key $\mathrm{PK_R}$, the cloud server's public key $\mathrm{PK_S}$, and the data owner's private key $\mathrm{SK_O}$ as input. Then, the data owner randomly selects two random numbers $r_1, r_2 \in \mathbb{Z}_q^*$. For each keyword $w_i \in W$, the following computation is performed to generate the ciphertext $C_w = (C_1, C_2, C_3)$:

$$\begin{aligned} K &= x\mathrm{PK_R} \\ C_1 &= r_1 P \\ &= H_2(w_1)P \\ C_2 &= r_2 H(\mathrm{PK_R}, \mathrm{PK_O}, w_i, K)x^{-1} \\ C_3 &= r_2 \mathrm{PK_R} + r_1 \mathrm{PK_S} \end{aligned} \tag{10}$$

  Finally, the data owner uploads the encrypted data file along with the $C_w = (C_1, C_2, C_3)$ to the cloud server.
- Trapdoor($\boldsymbol{w}, \mathrm{SK_R}, \mathrm{PK_O}, \mathrm{PK_R}, \mathrm{PK_S}$). Given params, the data owner's public key $\mathrm{PK_O}$, the cloud server's public key $\mathrm{PK_S}$, and the data owner's private key $\mathrm{SK_R}$ as input. The data recipient then randomly selects two random numbers $t_1, t_2 \in Z_q^*$. For keyword $w' \in W$, perform the following calculation to generate the keyword trapdoor $T_{w'} = (T_1, T_2, T_3)$.

$$\begin{aligned} K' &= y\mathrm{PK_O} \\ T_1 &= t_1 P \\ T_2 &= t_2 H(\mathrm{PK_{ID_R}}, \mathrm{PK_{ID_O}}, w', K')y^{-1} \\ T_3 &= t_2 \mathrm{PK_O} + t_1 \mathrm{PK_S} \end{aligned} \tag{11}$$

  Finally, the data recipient submits the keyword trapdoor $T_{w'} = (T_1, T_2, T_3)$ to the cloud server.
- Test(C_w, T_w). When the cloud server receives the keyword trapdoor $T_{w'}$ sent by the retrieving user, it performs a matching test against the encrypted keyword stored on the server. First, the cloud server uses the private key $\mathrm{SK_S}$ to compute and verify whether $T_2C_3 - sT_2C_1 = C_2T_3 - sC_2T_1$ holds true. If it does, this indicates that the keyword trapdoor matches the encrypted keyword. The cloud server then returns the encrypted data file to the data recipient. Otherwise, it returns 0.

3.2 Correctness

We will validate the correctness of the proposed scheme. First, there is

$$\begin{aligned} & C_2T_3 - sC_2T_1 = C_2(T_3 - sT_1) \\ &= C_2(t_2\mathrm{PK_O} + t_1\mathrm{PK_S} - st_1P) \\ &= C_2t_2\mathrm{PK_O} \\ &= r_2H(\mathrm{PK_R}, \mathrm{PK_O}, w_i, K)x^{-1}t_2\mathrm{PK_O} \\ &= r_2t_2H_1(\mathrm{PK_R}, \mathrm{PK_O}, w_i, K)P \end{aligned} \tag{12}$$

At the same time, it can be calculated

$$\begin{aligned} & T_2C_3 - sT_2C_1 = T_2(C_3 - sC_1) \\ &= T_2(r_2\mathrm{PK_R} + r_1\mathrm{PK_S} - sr_1P) \\ &= T_2r_2\mathrm{PK_R} \\ &= t_2H_1(\mathrm{PK_R}, \mathrm{PK_O}, w', K')y^{-1}r_2\mathrm{PK_R} \\ &= t_2r_2H(\mathrm{PK_R}, \mathrm{PK_0}, w', K')P \end{aligned} \tag{13}$$

Second

$$K = x\mathrm{PK_R} = xyP = \mathrm{yPK_O} = K' \tag{14}$$

if $w_i = w'$ then

$$H(\mathrm{PK_R}, \mathrm{PK_O}, w_i, K) = H(\mathrm{PK_R}, \mathrm{PK_O}, w', K') \tag{15}$$

That is, if $T_2C_3 - sT_2C_1 = C_2T_3 - sC_2T_1$ holds, then the construction of the proposed PAEKS scheme is correct.

4 Security Analysis

In this section, this paper will prove the security of the proposed scheme based on the security model defined earlier.

Theorem 1. The proposed scheme satisfies MCI security under the random oracle model if the CDH problem is hard.

Proof Suppose an adversary $\mathscr{A}$ can completely undermine MCI security with a non-negligible advantage in polynomial time. Then an algorithm B can be constructed to break the CDH security assumption. Given a CDH tuple (P, aP, bP), where $a, b \in \mathbb{Z}_q^*$ is unknown, B achieves this by engaging in the following game with adversary $\mathscr{A}$.

Setup. B runs the Setup algorithm and KeyGen algorithm to generate the system parameter $params = (\lambda, G, q, P, H)$. Then B returns $(\mathrm{params}, \mathrm{PK_O}, \mathrm{PK_R}, \mathrm{PK_S})$ to adversary $\mathscr{A}$.

Phase 1. For adversary $\mathscr{A}$, the following multiple polynomial adaptive queries can be executed.

1) Hash lookup. B maintains an empty list L_H containing a tuple $(\mathrm{PK'_R}, \mathrm{PK'_O}, w_i, k_i, h_i)$. If adversary $\mathscr{A}$ provides a tuple $(\mathrm{PK'_R}, \mathrm{PK'_O}, w_i, k_i)$ already stored in L_H, B retrieves h_i from L_H and returns it to adversary $\mathscr{A}$. Otherwise, B randomly selects $h_i \in Z_q^*$, sets $(\mathrm{PK'_R}, \mathrm{PK'_O}, w_i, k_i) = h_i$ to join L_H, and returns h_i to adversary $\mathscr{A}$.

2) Ciphertext Query. When adversary $\mathscr{A}$ queries the ciphertext for $(\mathrm{PK'_R}, w_i)$, B retrieves $(\mathrm{PK'_R}, \mathrm{PK_O}, w_i, k_i, h_i)$ from L_H.

Then, B randomly selects $r_{1i}, r_{2i} \in Z_q^*$ and computes

$$\begin{aligned} C_{1i} &= r_{1i}P \\ C_{2i} &= r_{2i}h_i\mathrm{SK_O}^{-1} \\ C_{3i} &= r_{2i}\mathrm{PK'_R} + r_{1i}\mathrm{PK_S} \end{aligned} \tag{16}$$

3) Trapdoor Query. When adversary $\mathscr{A}$ queries the ciphertext for $(\mathrm{PK'_O}, w_i)$, B retrieves $(\mathrm{PK_R}, \mathrm{PK'_O}, w_i, k_i, h_i)$ from L_H. Then, B randomly selects $t_{1i}, t_{2i} \in Z_q^*$ and computes

$$\begin{aligned} T_{1i} &= t_{1i}P \\ T_{2i} &= t_{2i}h_i\mathrm{SK_R}^{-1} \\ T_{3i} &= t_{2i}\mathrm{PK_O} + t_{1i}\mathrm{PK_S} \end{aligned} \tag{17}$$

Challenge: Adversary $\mathscr{A}$ selects two keyword pairs $w_0 = (w_{0,i}^*)_{i\in[1,n]}$ and $w_1 = (w_{1,i}^*)_{i\in[1,n]}$, where keywords such as $(\mathrm{PK_O}, w_{0,i}^*)$ and $(\mathrm{PK_O}, w_{1,i}^*)$ have not been queried for ciphertext. B randomly selects $\delta \in \{0,1\}$; for each $i \in [1,n]$, B chooses $r_{1i}^*, r_{2i}^* \in Z_q^*$, selects $h_i^* \in Z_q$, and computes $C_{w_{\delta,i}^*}$, where

$$\begin{aligned} C_{1i}^* &= r_{1i}^* P \\ C_{2i}^* &= r_{2i}^* h_i^* \mathrm{SK_O}^{-1} \\ C_{3i}^* &= r_{2i}^* \mathrm{PK_R} + r_{1i}^* \mathrm{PK_S} \end{aligned} \tag{18}$$

Finally, B outputs the challenge ciphertext $\{C_{w_{\delta,i}^*}\}_{i\in[1,n]}$ to adversary $\mathscr{A}$.

Phase 2. Similar to Phase 1, Adversary $\mathscr{A}$ may continue to perform polynomial-time adaptive queries. However, the restriction is that Adversary $\mathscr{A}$ may not query any challenge keywords from the challenge phase.

Guess. Adversary $\mathscr{A}$ outputs his guess $\delta' \in \{0,1\}$. If $\delta' = \delta$, then B sets $C_{1i}^*, \mathrm{PK_S} = bP$. Then $C_{3i}^* - (h_i^*)^{-1}\mathrm{SK_O}C_{2i}^*\mathrm{PK_R} = abP$ can be computed, meaning B can solve the CDH problem.

MCI ensures that even if an attacker is able to choose multiple plaintexts and observe the corresponding ciphertexts or indexes, the ciphertexts and indexes remain indistinguishable. Therefore, the attacker cannot infer the plaintext content by analyzing the frequency or statistical patterns of the ciphertext, effectively mitigating the risk of frequency analysis attacks. □

Theorem 2. The proposed scheme satisfies MTI security under the random oracle model if the CDH problem is hard.

Proof Suppose an adversary $\mathscr{A}$ can completely undermine MTI security with a non-negligible advantage in polynomial time. Then an algorithm B can be constructed to break the CDH security assumption. Given a CDH tuple (P, aP, bP), where $a, b \in \mathbb{Z}_q^*$ is unknown, B achieves this by engaging in the following game with adversary $\mathscr{A}$.

Setup. B runs the Setup algorithm and KeyGen algorithm to generate the system parameter $params = (\lambda, G, q, P, H)$. Then B returns $(\mathrm{params}, \mathrm{PK_O}, \mathrm{PK_R}, \mathrm{PK_S})$ to adversary $\mathscr{A}$.

Phase 1. For adversary $\mathscr{A}$, the following multiple polynomial adaptive queries can be executed.

1) Hash lookup. B maintains an empty list L_H containing a tuple $(\mathrm{PK'_R}, \mathrm{PK'_O}, w_i, k_i, h_i)$. If adversary $\mathscr{A}$ provides a tuple $(\mathrm{PK'_R}, \mathrm{PK'_O}, w_i, k_i)$ already stored in L_H, B retrieves h_i from L_H and returns it to adversary $\mathscr{A}$. Otherwise, B randomly selects $h_i \in Z_q^*$, sets $(\mathrm{PK'_R}, \mathrm{PK'_O}, w_i, k_i) = h_i$ to join L_H, and returns h_i to adversary $\mathscr{A}$.

2) Ciphertext Query. When adversary $\mathscr{A}$ queries the ciphertext for $(\mathrm{PK'_R}, w_i)$, B retrieves $(\mathrm{PK'_R}, \mathrm{PK_O}, w_i, k_i, h_i)$ from L_H. Then, B randomly selects $r_{1i}, r_{2i} \in Z_q^*$ and computes

$$\begin{aligned} C_{1i} &= r_{1i}P \\ C_{2i} &= r_{2i}h_i\mathrm{SK_O}^{-1} \\ C_{3i} &= r_{2i}\mathrm{PK'_R} + r_{1i}\mathrm{PK_S} \end{aligned} \tag{19}$$

3) Trapdoor Query. When adversary $\mathscr{A}$ queries the ciphertext for $(\mathrm{PK'_O}, w_i)$, B retrieves $(\mathrm{PK_R}, \mathrm{PK'_O}, w_i, k_i, h_i)$ from L_H. Then, B randomly selects $t_{1i}, t_{2i} \in Z_q^*$ and computes

$$\begin{aligned} T_{1i} &= t_{1i}P \\ T_{2i} &= t_{2i}h_i\mathrm{SK_R}^{-1} \\ T_{3i} &= t_{2i}\mathrm{PK_O} + t_{1i}\mathrm{PK_S} \end{aligned} \tag{20}$$

Challenge: Adversary $\mathscr{A}$ selects two keyword pairs $w_0 = (w_{0,i}^*)_{i\in[1,n]}$ and $w_1 = (w_{1,i}^*)_{i\in[1,n]}$, where keywords such as $(\mathrm{PK_O}, w_{0,i}^*)$ and $(\mathrm{PK_O}, w_{1,i}^*)$ have not been queried for ciphertext. B randomly selects $\delta \in \{0,1\}$; for each $i \in [1,n]$, B chooses $t_{1i}^*, t_{2i}^* \in Z_q^*$, selects $h_i^* \in Z_q$, and computes $T_{w_{\delta,i}^*}$, where

$$\begin{aligned} T_{1i}^* &= t_{1i}^* P \\ T_{2i}^* &= t_{2i}^* h_i^* \mathrm{SK_R}^{-1} \\ T_{3i}^* &= t_{2i}^* \mathrm{PK_O} + t_{1i}^* \mathrm{PK_S} \end{aligned} \tag{21}$$

Finally, B outputs the challenge ciphertext $\{T_{w_{\delta,i}^*}\}_{i\in[1,n]}$ to adversary $\mathscr{A}$.

Phase 2. Similar to Phase 1, Adversary $\mathscr{A}$ may continue to perform polynomial-time adaptive queries. However, the restriction is that Adversary $\mathscr{A}$ may not query any challenge keywords from the challenge phase.

Guess. Adversary $\mathscr{A}$ outputs his guess $\delta' \in \{0,1\}$. If $\delta' = \delta$, then B sets $C_{1i}^*, \mathrm{PK_S} = bP$. Then $T_{3i}^* - (h_i^*)^{-1}\mathrm{SK_R}T_{2i}^*\mathrm{PK_O} = abP$ can be computed, meaning B can solve the CDH problem.

MTI ensures that even if an attacker can choose multiple ciphertexts and observe the corresponding trapdoors, these trapdoors remain indistinguishable. Therefore, the attacker cannot obtain any information about the plaintext or the encryption process from the trapdoors, nor can they infer the plaintext content by analyzing the frequency or statistical characteristics of the trapdoors, effectively mitigating the risk of frequency analysis attacks. □

5 Performance Analysis

In this section, we analyze the performance of six comparable schemes. Among them, [10, 19–21] are pairing-based searchable encryption schemes, while [13] and [16] are pairing-free searchable encryption schemes.The experiments were conducted on a Raspberry Pi 3 Model B equipped with a quad-core ARM Cortex-A72 processor and 1 GB of memory, and the implementation was based on the Charm-Crypto 0.50 lib-

Table 1: Running time of basic operations

Symbol	Description	Time(ms)
T_G	Running time of map to point G	0.771
T_{pm}	Running time of point scalar multiplication	7.254
T_{bp}	Running time of bilinear pairing	14.875
$T_{\exp_{G_1}}$	Running time of exponentiation operation in G_1	10.039
$T_{\exp_{G_2}}$	Running time of exponentiation operation in G_2	10.043
$T_{\exp_{G_T}}$	Running time of exponentiation operation in G_T	1.825
T_{G_1}	Running time of map to point G_1	23.305
T_h	Running time of general hash function	0.011

-rary. Two representative schemes were considered: a pairing-free searchable encryption scheme instantiated on the secp160r2 elliptic curve, where group elements are 320 bits in size and the scalar field $\mathbb{Z}_q$ has a bit length of 160; and a pairing-based scheme built on the SS512 curve, where the source group $\mathbb{G}_1$ and the target group $\mathbb{G}_2$ have element sizes of 512 bits and 1024 bits, respectively. The SS512 curve is a supersingular elliptic curve over a prime field $\mathbb{F}_p$ defined by $y^2 = x^3 + x$ with an embedding degree of $k = 2$. In addition, all general-purpose cryptographic functions are implemented using SHA-256. Table 1 summarizes the notations and average execution times of the core operations, including scalar multiplication, bilinear pairing, exponentiation in various groups, map-to-point operations, and the general hash function.

5.1 Computation Cost

According to the data in Table 2 and Figure 2, our scheme significantly outperforms pairing-based schemes in computational overhead. Specifically, in the ciphertext generation and trapdoor generation phases, our scheme consumes 29.027 ms, which is 20%, 45.8%, and 65.3% lower than the 36.314 ms, 53.422 ms, and 83.55 ms of [16, 19, 21], respectively. In the testing algorithm, our scheme consumes 29.016 ms, slightly higher than some pairing-free schemes (7.254 ms in [13] and 14.886 ms in [20]), but still significantly lower than the pairing-based schemes (59.5 ms in [21] and 29.75 ms in [10]), reducing the time by 51.2% and 2.5%, respectively. In terms of total time, our scheme takes 87.07 ms, which is only higher than the 50.833 ms of [13]. However, despite having the lowest overhead, scheme [13] uses a deterministic trapdoor generation algorithm, which makes it vulnerable to frequency analysis attacks. In contrast, our scheme provides stronger frequency analysis protection, preventing attackers from inferring keywords from the ciphertext, which is crucial in CIoT environments.

While pairing-free schemes like [13] and [20] offer certain computational efficiency advantages, they are vulnerable to frequency analysis attacks. Our scheme, however, ensures high security while maintaining superior computational efficiency, achieving a balance between data privacy protection and computational efficiency in CIoT environments.

5.2 Communication Cost

This paper compares the communication overhead of our proposed scheme with six other schemes, analyzing it across three stages: public key generation, ciphertext generation, and trapdoor generation. As shown in Table 3 and Figure 3, during the public key generation stage, pairing-based schemes (involving $\mathbb{G}_1$ group elements) generally incur higher communication overhead. For example, schemes [16, 19] generate public keys of 640 bits and 512 bits, respectively, while more complex schemes, such as [10, 21], generate larger public keys and ciphertexts. In particular, scheme [21] has a ciphertext size of 2048 bits, making it the highest in communication overhead among all the schemes.

In contrast, the non-pairing scheme proposed in this paper significantly reduces communication overhead. Our scheme does not rely on $\mathbb{G}_1$ group elements, with the public key size being 320 bits and both the ciphertext and trapdoor sizes being 800 bits, providing a clear advantage in communication overhead compared to pairing-based schemes. Particularly during ciphertext generation, the communication cost of our proposed scheme is significantly lower than that of pairing-based schemes. For instance, scheme [20] generates a ciphertext of 1024 bits, whereas our scheme only requires 800 bits.In the trapdoor generation stage, pairing-based schemes typically produce larger trapdoor sizes. For example, schemes [10, 19] have trapdoor sizes of 1024 bits and 2048 bits, respectively, while the trapdoor size of our proposed non-pairing scheme is 800 bits, further demonstrating

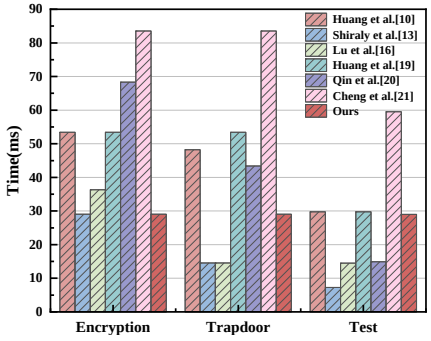

Figure 2: Comparison of computational complexity among different algorithms in the solution.

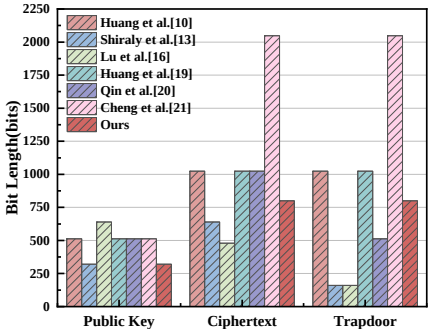

Figure 3: Communication cost of each scheme.

Table 2: Comparison of different PAEKS schemes

Schemes	Encryption	Trapdoor	Test	Frequency analysis resistance
Huang et al.[10]	$T_G + 3T_{expG}$	$T_G + T_{expG} + T_{bp}$	$2T_b$	×
Shirali et al.[13]	$4T_{pm} + 3T_h$	$2T_{pm} + 2T_h$	T_{pm}	×
Lu et al.[16]	$5T_{pm} + 4T_h$	$2T_{pm} + 3T_h$	$2T_{pm} + T_h$	×
Huang et al.[19]	$T_G + 3T_{expG}$	$T_G + 3T_{expG}$	$2T_G + T_{expG}$	×
Qin et al.[20]	$T_G + T_h + 3T_{expG} + T_{bp}$	$T_G + T_h + 2T_{expG}$	$T_{bp} + T_h$	×
Cheng et al.[21]	$T_G + T_h + 6T_{expG}$	$T_G + T_h + 6T_{expG}$	$4T_b$	✓
Ours	$4T_{pm} + T_h$	$4T_{pm} + T_h$	$4T_{pm}$	✓

the advantage in communication overhead. Although the overhead of our proposed scheme is slightly higher than some lightweight non-pairing schemes, this is due to the introduction of additional group elements and scalars to enhance security and resist frequency analysis attacks.Therefore, the non-pairing scheme proposed in this paper significantly reduces communication overhead while ensuring strong security, making it highly suitable for resource- and communication-constrained CIoT environments.

Table 3: Communication overhead comparison

Schemes	Size of PK(bits)	Size of CT(bits)	Size of TR(bits)
Huang et al.[10]	$\|G_1\| = 512$	$2\|G_1\| = 1024$	$\|G_T\| = 1024$
Shiraly et al.[13]	$\|G\| = 320$	$2\|G\| = 640$	$\|Z_q^*\| = 160$
Lu et al.[16]	$2\|G\| = 640$	$\|G\| + h = 480$	$h = 160$
Huang et al.[19]	$\|G_T\| = 512$	$2\|G_1\| = 1024$	$2\|G_1\| = 1024$
Qin et al.[20]	$\|G_1\| = 512$	$2\|G_1\| = 1024$	$\|G_1\| = 512$
Cheng et al.[21]	$\|G_1\| = 512$	$4\|G_1\| = 2048$	$4\|G_1\| = 2048$
Ours	$\|G\| = 320$	$\|G\| + \|Z_q^*\| = 800$	$2\|G\| + \|Z_q^*\| = 800$

6 Conclusion

The rapid development of cloud-assisted IoT has significantly transformed data management and access in industrial automation, healthcare, and smart grids, while posing challenges for secure and efficient search over encrypted data. Although numerous PEKS/PAEKS variants had been proposed, these methods in resource-constrained CIoT environments still suffered from the high computational cost of bilinear pairings and security risks from frequency-analysis attacks. To address these pain points, this paper proposed a pairing-free PAEKS scheme based on elliptic-curve scalar multiplication. By eliminating bilinear pairings and adopting a probabilistic trapdoor design, the scheme strengthened resistance to frequency-analysis attacks and achieved a more favorable balance between security and efficiency. Evaluation results showed that, under comparable security levels, the proposed scheme effectively reduced computational and communication overhead, demonstrating strong feasibility for cloud-assisted IoT deployment and promising application prospects.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 62272123; Project of High-level Innovative Talents of Guizhou Province under Grant [2020]6008; Science and Technology Program of Guizhou Province under Grant [2020]5017, [2022]065; Science and Technology Program of Guiyang under Grant [2021]1-5 and [2022]2-4.

Author Contributions

Conceptualization, Kunlong Jin and Kaiqi Liu; methodology, Kunlong Jin and Kaiqi Liu; software, Kunlong Jin and Shuai Wang; validation, Kunlong Jin and Kaiqi Liu; formal analysis, Kaiqi Liu and Lujia Shi; investigation, Lujia Shi; resources, Jiaen Zhou; data curation, Kunlong Jin; writing—original draft preparation, Kunlong Jin; writing—review and editing, Jiaen Zhou; visualization, Kunlong Jin; supervision, Jiaen Zhou and Shuai Wang; project administration, Jiaen Zhou and Shuai Wang; funding acquisition, Shuai Wang. All authors have read and agreed to the published version of the manuscript.

Conflict of Interest

All the authors declare that they have no conflict of interest.

References

[1] Andola, N., Gahlot, R., Yadav, V.K., Venkatesan, S., Verma, S.: Searchable encryption on the cloud: a survey. The Journal of Supercomputing **78**(7), 9952–9984 (2022) https://doi.org/10.1007/s11227-022-04309-6

[2] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, pp. 44–55 (2000). https://doi.org/10.1109/SECPRI.2000.848445

[3] Li, J., Huang, Y., Wei, Y., Lv, S., Liu, Z., Dong, C., Lou, W.: Searchable symmetric encryption with forward search privacy. IEEE Transactions on Dependable and Secure Computing **18**(1), 460–474 (2019) https://doi.org/10.1109/TDSC.2019.2894411

[4] Wang, Q., Zhang, X., Qin, J., Ma, J., Huang, X.: A verifiable symmetric searchable encryption scheme based on the avl tree. The Computer Journal **66**(1), 174–183 (2023) https://doi.org/10.1093/comjnl/bxab152

[5] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In International Conference on the Theory and Applications of Cryptographic Techniques, pp. 506–522 (2004). https://doi.org/10.1007/978-3-540-24676-3_30

[6] Liang, K., Susilo, W.: Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Transactions on Information Forensics and Security **10**(9), 1981–1992 (2015) https://doi.org/10.1109/TIFS.2015.2442215

[7] Xia, Z., Wang, X., Sun, X., Wang, Q.: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE transactions on parallel and distributed systems **27**(2), 340–352 (2015) https://doi.org/10.1109/TPDS.2015.2401003

[8] Byun, J.W., Rhee, H.S., Park, H.-A., Lee, D.H.: Offline keyword guessing attacks on recent keyword search schemes over encrypted data. In Workshop on Secure Data Management, pp. 75–83 (2006). https://doi.org/10.1007/11844662_6

[9] Hu, C., Liu, P.: A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension. In International Conference on Computer Science, Environment, Ecoinformatics, and Education, pp. 131–136 (2011). https://doi.org/10.1007/978-3-642-23324-1_23

[10] Huang, Q., Li, H.: An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Information Sciences **403–404**, 1–14 (2017) https://doi.org/10.1016/j.ins.2017.03.038

[11] Lu, Y., Li, J., Zhang, Y.: Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks. IEEE Transactions on Services Computing **14**(6), 2041–2054 (2019) https://doi.org/10.1109/TSC.2019.2910113

[12] Karati, A., Fan, C.-I., Zhuang, E.-S.: Reliable data sharing by certificateless encryption supporting keyword search against vulnerable kgc in industrial internet of things. IEEE Transactions on Industrial Informatics **18**(6), 3661–3669 (2021) https://doi.org/10.1109/TII.2021.3112986

[13] Shiraly, D., Pakniat, N., Noroozi, M., Eslami, Z.: Pairing-free certificateless authenticated encryption with keyword search. Journal of Systems Architecture **124**, 102390 (2022) https://doi.org/10.1016/j.sysarc.2021.102390

[14] Liu, X., Sun, Y., Dong, H.: A pairing-free certificateless searchable public key encryption scheme for iomt. Journal of Systems architecture **139**, 102885 (2023) https://doi.org/10.1016/j.sysarc.2023.102885

[15] Li, J., Wang, M., Lu, Y., Zhang, Y., Wang, H.: Abksskga: Attribute-based keyword search secure against keyword guessing attack. Computer Standards & Interfaces **74**, 103471 (2021) https://doi.org/10.1016/j.csi.2020.103471

[16] Lu, Y., Li, J.: 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 (2021) https://doi.org/10.1109/TMC.2021.3077508

[17] Liu, X., Dong, H., Kumari, N., Kar, J.: A pairing-free certificateless searchable public key encryption scheme for industrial internet of things. IEEE Access **11**, 58754–58764 (2023) https://doi.org/10.1109/ACCESS.2023.3285114

[18] Cheng, L., Qin, J., Meng, F.: Privacy leakage of certificateless public key authenticated searchable encryption via frequency analysis: Attacks and revises. Computer Standards & Interfaces **87**, 103762 (2024) https://doi.org/10.1016/j.csi.2023.103762

[19] Huang, Q., Huang, P., Li, H., Huang, J., Lin, H.: A more efficient public-key authenticated encryption scheme with keyword search. Journal of Systems Architecture **137**, 102839 (2023) https://doi.org/10.1016/j.sysarc.2023.102839

[20] Qin, B., Cui, H., Zheng, X., Zheng, D.: Improved security model for public-key authenticated encryption with keyword search. In International Conference on Provable Security, pp. 19–38 (2021). https://doi.org/10.1007/978-3-030-90402-9_2

[21] Cheng, L., Qin, J., Feng, F., Meng, F.: Security-enhanced public-key authenticated searchable encryption. Information Sciences **647**, 119454 (2023) https://doi.org/10.1016/j.ins.2023.119454