On the Security of Public Key Authenticated Encryption with Keyword Search with Sender-independent Search Complexity*

Takeshi Yoshida[1] and Keita Emura[†1,2]

[1.]Kanazawa University, Japan.
[2.]AIST, Japan.

August 31, 2026

Abstract

Li et al. (IEEE Transactions on Dependable and Secure Computing 2026) proposed proxy-free public key authenticated encryption with ciphertext update and keyword search (proxy-free PAUKS). In this short note, we demonstrate that keyword information is leaked from updated ciphertexts. We also demonstrate that our attack is effective against the PAUKS scheme proposed by Li et al. (IEEE Transactions on Information Forensics and Security 2023).

1 Introduction

Keyword guessing attack (KGAs) are fundamental issues in public key Encryption with Keyword Search (PEKS) [1] where keyword information is leaked from trapdoors. Public key Authenticated Encryption with Keyword Search (PAEKS) [6,9] is a variant of PEKS that protects against KGAs. The main idea is that a sender (encryptor) generates a ciphertext of a keyword by using not only the receiver's public key but also the sender's secret key. At the expense of KGA resistance, PAEKS provides an inefficient search complexity where it requires a receiver to generate trapdoors for all senders, i.e., the communication costs and searching time are linearly proportional to the number of senders.

Li et al. [7] proposed public key authenticated encryption with ciphertext update and keyword search (PAUKS) to address this issue. Briefly, the encryption procedure is the same as that of PAEKS where a sender generates a ciphertext for a keyword using the sender's secret key and a receiver's public key. As a difference, two types of trapdoors are generated by a receiver: one that depends on a sender, as in PAEKS, and another that is sender-independent, as in PEKS. Using an update key, which is generated by a sender's public key and a receiver's secret key, the sender-related information in the ciphertext is removed via the ciphertext updating procedure. Then, the search complexity for the sender-independent ciphertext using the second type of trapdoor becomes independent of the number of senders, as in PEKS. Because a fully trusted proxy, that updates ciphertexts, is required in PAUKS, Li et al. [8] proposed a proxy-free PAUKS scheme by modifying the Li et al. PAUKS scheme [7].

*The main content of this paper appears IEICE transactions on fundamentals of electronics communications and computer sciences, with the title: Security Analysis on "Proxy-Free Public-Key Authenticated Updatable and Searchable Encryption for Cloud Storage".

[†]Corresponding Author: k-emura@se.kanazawa-u.ac.jp

Our Contribution. In this short note, we show that keyword information can be leaked from ciphertexts in the proxy-free PAUKS scheme [8]. We emphasize that our attack follows the security model given by the original paper [8]. We also demonstrate that our attack is effective against the PAUKS scheme [7].

We focus on a part of a ciphertext that is uniquely determined when the sender, receiver, and encrypted keyword are fixed. This fact allows us to check whether two ciphertexts are encryptions of the same keyword. Therefore, an adversary sends the challenge keyword w_0^* to the encryption oracle in [7] and the updated encryption oracle in [8], and checks whether the challenge ciphertext is encryption of w_0^* that breaks the security of PAUKS.

Related Work. In addition to [7,8], PAEKS schemes with sender-independent search complexity have been proposed [3,11–13,16]. Here, we briefly introduce vulnerabilities of these PAEKS schemes, except [13].

- Zhang et al. [16] proposed Multi-User Keyword-Field-Free Conjunctive Searchable Encryption (MU-KFFCSE). In [5], a vulnerability of the MU-KFFCSE scheme has been reported where keyword information is leaked from trapdoors, and an adversary can complete the attack without obtaining any secret value. Moreover, a flaw of the security proof is also reported where an instance of the underlying complexity assumption is independent of a keyword-related part of a ciphertext and thus the assumption does not contribute to hiding keyword information.

- Wang et al. [11] proposed Identity-Based Group Encryption with Keyword Search (IBGEKS) where each sender belongs to a group and a trapdoor is designated for a group, to reduce the search complexity in PAEKS. In [14], a vulnerability of the IBGEKS scheme has been reported where keyword information is leaked from trapdoors. In the IBGEKS scheme, a trapdoor for a keyword can be seen as a Boneh-Lynn-Shacham (BLS) signature [2] and a verification key of the BLS signature scheme can be obtained via a sender's secret key or a trapdoor. Thus, keyword information can be leaked from trapdoors via BLS signature verification.

- Xu et al. [12] proposed a lightweight multi-user PAEKS (LM-PAEKS) scheme. In [4], a vulnerability of the LM-PAEKS scheme has been reported where keyword information can be leaked from ciphertexts due to the fact that two ciphertexts are linkable, meaning that anyone can determine whether they correspond to the same keyword. One straightforward fix is to restrict the encryption query such that an adversary is not allowed to send the challenge keywords to the encryption oracle, which model is so-called CI (ciphertext indistinguishability) security. However, CI security is weak in the sense that one keyword is allowed to be encrypted once. This one-time use keyword does not match the actual application of PAEKS. Thus, full CI security, where an adversary is allowed to send the challenge keywords to the encryption oracle, should be provided.

- Cheng and Meng [3] proposed a server-aided PAEKS (SA-PAEKS) scheme which adds a sender server and a receiver server. A sender generates a ciphertext associated with both servers, and a receiver generates a trapdoor similarly. The sender server then transforms the ciphertext, and the receiver server removes receiver-related information, producing a trapdoor associated only with the two servers. Then, the search complexity does not depend on the number of senders. In [13], a vulnerability of the SA-PAEKS scheme has been reported where the cloud server (tester) can obtain keyword information from both a ciphertext and a trapdoor. By using the cloud server's secret key, whether two ciphertexts are encryptions of the same keyword can be checked. In [15], a general attack has been reported. An adversary,

modeled as a cloud server, generates a sender's key and a receiver's key, respectively, and runs the encryption algorithm and the trapdoor algorithm using these keys and the challenge keyword. Since they are independent of sender and receiver, they can be used for running the test algorithm with the challenge trapdoor and the challenge ciphertext, respectively.

2 Definition of Proxy-Free PAUKS

We present the definition of Proxy-Free PAUKS [8].

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$ as input, the setup algorithm outputs the common parameter pp.
- $\mathsf{KeyGen_R}(\mathsf{pp})$: The receiver key generation algorithm takes pp, and outputs a key pair $(\mathsf{pk_R}, \mathsf{sk_R})$.
- $\mathsf{KeyGen_S}(\mathsf{pp})$: The sender key generation algorithm takes pp, and outputs a key pair $(\mathsf{pk_S}, \mathsf{sk_S})$.
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk_S}, \mathsf{pk_R}, w)$: The encryption algorithm takes pp, $\mathsf{sk_S}$, $\mathsf{pk_R}$, and a keyword w, and outputs a ciphertext C.
- $\mathsf{Trapdoor}(\mathsf{pp}, \mathsf{sk_R}, \mathsf{pk_S}, w')$: The trapdoor generation algorithm takes pp, $\mathsf{sk_R}$, $\mathsf{pk_S}$, and a keyword w', and outputs a trapdoor $T_{w'}$.
- $\mathsf{Test}(\mathsf{pp}, \mathsf{pk_S}, C, T_w)$: The test algorithm takes pp, $\mathsf{pk_S}$, C, and T_w generated with $\mathsf{pk_S}$, and outputs 1 if C and T_w are generated from the same keyword, and 0 otherwise.
- $\mathsf{UKG}(\mathsf{pp})$: The update key generation algorithm takes pp, and outputs an update key uk.
- $\mathsf{TokG}(\mathsf{pp}, \mathsf{pk_S}, \mathsf{uk_R}, \mathsf{sk_R})$: The update token generation algorithm takes pp, $\mathsf{pk_S}$, $\mathsf{uk_R}$ (an update key generated by R), and $\mathsf{sk_R}$, and outputs an update token $\Delta_{\mathsf{S,R}}$.
- $\mathsf{UpdEnc}(\mathsf{pp}, C, \Delta_{\mathsf{S,R}}, \mathsf{pk_S})$: The ciphertext update algorithm takes pp, C, $\Delta_{\mathsf{S,R}}$, and $\mathsf{pk_S}$, and outputs an updated ciphertext $\bar{C}$.
- $\mathsf{ConstTrapdoor}(\mathsf{pp}, \mathsf{sk_R}, w')$: The constant trapdoor generation algorithm takes pp, $\mathsf{sk_R}$, and a keyword to be searched w', and outputs a constant trapdoor $\bar{T}_{w'}$.
- $\mathsf{UpdTest}(\mathsf{pp}, \bar{C}, \bar{T}_w)$: The test algorithm for updated ciphertexts and constant trapdoors takes pp. $\bar{C}$, and $\bar{T}_w$, and outputs 1 if $\bar{C}$, and $\bar{T}_w$ contain the same keyword, and 0 otherwise.

Updated Ciphertext Indistinguishability [8]. We introduce the security model that guarantees no keyword information is revealed from updated ciphertexts (Definition 7 in [8]). Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{KeyGen_R}(\mathsf{pp})$, $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{KeyGen_S}(\mathsf{pp})$, and $\mathsf{uk_R} \leftarrow \mathsf{UKG}(\mathsf{pp})$, and sends $(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S})$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to issue the following queries.

- Ciphertext Query: Given a keyword w, return $C \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{sk_S}, \mathsf{pk_R}, w)$.
- Updated Ciphertext Query: Given a keyword w, return $\bar{C} \leftarrow \mathsf{UpdEnc}(\mathsf{pp}, C, \Delta_{\mathsf{S,R}}, \mathsf{pk_S})$ where $C \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{sk_S}, \mathsf{pk_R}, w)$ and $\Delta_{\mathsf{S,R}} \leftarrow \mathsf{TokG}(\mathsf{pp}, \mathsf{pk_S}, \mathsf{uk_R}, \mathsf{sk_R})$.
- Trapdoor Query: Given a keyword w, return $T_w \leftarrow \mathsf{Trapdoor}(\mathsf{pp}, \mathsf{sk_R}, \mathsf{pk_S}, w)$.

We omit other queries irrelevant to our attack. $\mathcal{A}$ declares two challenge keywords, w_0^* and w_1^* which have not been queried for trapdoors and for ciphertexts. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$ and sends the challenge ciphertext $C^* \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{sk_S}, \mathsf{pk_R}, w_b^*)$ and the challenge updated ciphertext $\bar{C}^* \leftarrow \mathsf{UpdEnc}(\mathsf{pp}, C^*, \Delta_{\mathsf{S},\mathsf{R}}, \mathsf{pk_S})$ to $\mathcal{A}$. $\mathcal{A}$ is not allowed to send w_0^* and w_1^* as trapdoor and ciphertext queries. We emphasize that $\mathcal{A}$ is allowed to query w_0^* and w_1^* for updated ciphertexts. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins if $b = b'$. The advantage is defined as $|\Pr[b = b'] - 1/2|$. We emphasize that Li et al. wrote

"*Return an updated ciphertext $\bar{C}$ taking as input a keyword-sender-receiver tuple*"

as the definition of Updated Ciphertext Query, i.e., $\mathcal{A}$ does not send a ciphertext C as an updated ciphertext query. Moreover, $\mathcal{A}$ is not prohibited to send the challenge keywords as an updated ciphertext query (Definition 5 in [8] that gave definitions of queries).

3 Li et al. Proxy-Free PAUKS scheme

Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be groups with prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing. $H : \{0,1\}^* \to \mathbb{G}$, $H_1 : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$, and $H_2 : \mathbb{G} \to \mathbb{Z}_p$ are hash functions.

- $\mathsf{Setup}(1^\lambda)$: Output $\mathsf{pp} = (\mathbb{G}, \mathbb{G}_T, p, g, e, H, H_1, H_2)$.
- $\mathsf{KeyGen_R}(\mathsf{pp})$: Choose $y_1, y_2, y_3 \xleftarrow{\$} \mathbb{Z}_p$, compute $Y_1 = g^{y_1}$, $Y_2 = g^{y_2}$, and $Y_3 = g^{y_3}$, and output $(\mathsf{pk_R}, \mathsf{sk_R}) = ((Y_1, Y_2, Y_3), (y_1, y_2, y_3))$.
- $\mathsf{KeyGen_S}(\mathsf{pp})$: Choose $x \xleftarrow{\$} \mathbb{Z}_p$, compute $X = g^x$, and output $(\mathsf{pk_S}, \mathsf{sk_S}) = (X, x)$.
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{sk_S}, \mathsf{pk_R}, w)$: Choose $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$, compute $C_1 = g^{r_1}$, $C_2 = (Y_2^{H_1(Y_1^x, w)} \cdot Y_3)^{r_1}$, $C_3 = (Y_2^{H_2(Y_1^x)} \cdot Y_3)^{r_2}$, and $C_4 = H(w)^{r_2}$, and output $C = (C_1, C_2, C_3, C_4)$.
- $\mathsf{Trapdoor}(\mathsf{pp}, \mathsf{sk_R}, \mathsf{pk_S}, w')$: Choose $r_3 \xleftarrow{\$} \mathbb{Z}_p$, compute $T_1 = g^{r_3}$ and $T_2 = g^{\frac{r_3}{y_2 \cdot H1(X^{y_1}, w') + y_3}}$, and output $T_{w'} = (T_1, T_2)$.
- $\mathsf{Test}(\mathsf{pp}, \mathsf{pk_S}, C, T_w)$: Output 1 if $e(C_1, T_1) = e(C_2, T_2)$ holds, and 0 otherwise.
- $\mathsf{UKG}(\mathsf{pp})$: Choose $\mu, \nu \xleftarrow{\$} \mathbb{Z}_p$ and output $\mathsf{uk} = (\mu, \nu)$.
- $\mathsf{TokG}(\mathsf{pp}, \mathsf{pk_S}, \mathsf{uk_R}, \mathsf{sk_R})$: Compute $\Delta_{\mathsf{S},\mathsf{R}} = ((y_2 \cdot H_2(X^{y_1}) + y_3) - \nu)/(\mu(y_2 \cdot H_2(X^{y_1}) + y_3))$ and output $\Delta_{\mathsf{S},\mathsf{R}}$.
- $\mathsf{UpdEnc}(\mathsf{pp}, C, \Delta_{\mathsf{S},\mathsf{R}}, \mathsf{pk_S})$: Set $\bar{C}_1 = C_1$, $\bar{C}_2 = C_2$, $\bar{C}_3 = C_3$, and $\bar{C}_4 = C_4$, and compute $\bar{C}_5 = C_3^{\Delta_{\mathsf{S},\mathsf{R}}}$ and output $\bar{C} = (\bar{C}_1, \bar{C}_2, \bar{C}_3, \bar{C}_4, \bar{C}_5)$.
- $\mathsf{ConstTrapdoor}(\mathsf{pp}, \mathsf{sk_R}, w')$: Choose $r_4 \xleftarrow{\$} \mathbb{Z}_p$, compute $\bar{T}_1 = H(w')^{r_4}$, $\bar{T}_2 = H(w')^{\mu \cdot r_4}$, and $\bar{T}_3 = g^{\nu \cdot r_4}$, and output $\bar{T}_{w'} = (\bar{T}_1, \bar{T}_2, \bar{T}_3)$.
- $\mathsf{UpdTest}(\mathsf{pp}, \bar{C}, \bar{T}_w)$: Output 1 if $e(\bar{C}_3, \bar{T}_1) = e(\bar{C}_5, \bar{T}_2) e(\bar{C}_4, \bar{T}_3)$ holds, and 0 otherwise.

4 Our Attack

We focus on C_1 and C_2 that are contained in both an updated ciphertext and a ciphertext in [8]:

$$\bar{C}_1 = C_1 = g^{r_1} \text{ and } \bar{C}_2 = C_2 = (Y_2^{H_1(Y_1^x, w)} \cdot Y_3)^{r_1}$$

where $\mathsf{sk}_\mathsf{S} = x$ and $\mathsf{pk}_\mathsf{R} = (Y_1, Y_2, Y_3)$. Here, Y_1^x is a Diffie-Hellman key that is (non-interactively) shared between the sender S and the receiver R. We point out that $(Y_2^{H_1(Y_1^x, w)} \cdot Y_3)$ is uniquely determined if the sender, the receiver, and the keyword are fixed. Concretely, if two updated ciphertexts $\bar{C}^{(1)} = (\bar{C}_1^{(1)}, \bar{C}_2^{(1)}, \bar{C}_3^{(1)}, \bar{C}_4^{(1)}, \bar{C}_5^{(1)})$ and $\bar{C}^{(2)} = (\bar{C}_1^{(2)}, \bar{C}_2^{(2)}, \bar{C}_3^{(2)}, \bar{C}_4^{(2)}, \bar{C}_5^{(2)})$ are encryption of the same keyword, then

$$e(\bar{C}_1^{(1)}, \bar{C}_2^{(2)}) = e(\bar{C}_1^{(2)}, \bar{C}_2^{(1)})$$

holds. Thus, $\mathcal{A}$ sends w_0^* as an updated ciphertext query, and checks the above equation with the challenge ciphertext. If the equation holds, $\mathcal{A}$ outputs 0 (meaning that the challenge ciphertext is an encryption of w_0^*), and 1 (meaning that the challenge ciphertext is an encryption of w_1^*) otherwise.

Because a ciphertext of the PAUKS scheme [7] contains (C_1, C_2) with the same form, the same attack holds in the same manner, where $\mathcal{A}$ sends a ciphertext query w_0^*. We emphasize that an adversary is allowed to send the challenge keywords as ciphertext queries in the security model (See Definition 1 [7]), unlike to the security model in [8].

5 Discussion

Practical Issues Revealed by Our Attack. First, our attack demonstrates that, given two ciphertexts, regardless of whether they are before or after an update, it is possible to determine whether they correspond to the same keyword. That is, information indicating that two ciphertexts encrypt the same keyword is leaked even without performing a search, which constitutes a critical flaw for a searchable encryption scheme. We would like to clarify the impact of this vulnerability. In a typical searchable encryption setting, it is assumed that the scheme is used in conjunction with a separate encryption scheme for encrypting documents. Specifically, keywords associated with a document are encrypted using a searchable encryption scheme, and the resulting ciphertexts are stored together with the ciphertext of the document. After receiving a trapdoor from a receiver, the server performs a search using the trapdoor and returns the ciphertexts of the documents corresponding to the matching searchable ciphertexts. Through this process, the receiver obtains the documents associated with the queried keywords. In many practical scenarios, related documents naturally correspond to the same keyword. For example, medical documents concerning the same disease are typically associated with the same keyword (i.e., the disease name). Now consider a scenario in which it is possible to determine, without performing any search, that two searchable ciphertexts correspond to the same keyword. In this case, even though the documents themselves remain encrypted, it leaks that the corresponding documents are related to each other.

It is possible to avoid our attack by weakening the security model. For example, one could disallow the challenge keyword as input to ciphertext update queries. Alternatively, by taking the inputs of the ciphertext update algorithm into account, one could modify the model so that the adversary's ciphertext update queries include a ciphertext and only return the part of the ciphertext that is modified by the update, which corresponds to $\bar{C}_5$ in the proxy-free PAUKS scheme. Since the adversary is not allowed to submit the challenge keyword as a ciphertext query, the adversary cannot obtain any (updated) ciphertext of the challenge keyword other than the challenge ciphertext

itself. However, such weakened security models suffer from clear practical limitations. In particular, they only capture security in settings where each keyword is encrypted at most once. In practice, it is natural for a keyword to appear in multiple documents. If encrypting a keyword more than once is disallowed, the same ciphertext must be reused, which again leaks information indicating that the ciphertexts correspond to the same keyword, even without performing a search. From this perspective, we argue that weakening the security model merely to "apparently" prevent our attack does not address any realistic information-leakage issues in practice. To guarantee that no information about keywords is leaked from ciphertexts, adversaries should be given ciphertexts for arbitrary keywords, as well as the results of updating the challenge ciphertext.

Discussion of Possible Modifications and Their Limitations. Our attack basically utilizes the DDH test via symmetric pairings [10]. It is reasonable to consider replacing the symmetric pairing with an asymmetric pairing that realizes DDH-hard groups in order to achieve resistance to the proposed attack. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric pairing. Let us assign $C_1, C_2 \in \mathbb{G}_1$ and $T_1, T_2 \in \mathbb{G}_2$. Then, the modification does not affect the equation in the Test algorithm: $e(C_1, T_1) = e(C_2, T_2)$. We note, however, that Li et al. [8] mentioned that for two non-updated ciphertexts $C^{(1)} = (C_1^{(1)}, C_2^{(1)}, C_3^{(1)}, C_4^{(1)})$ and $C^{(2)} = (C_1^{(2)}, C_2^{(2)}, C_3^{(2)}, C_4^{(2)})$, if $e(C_3^{(1)}, C_4^{(2)}) = e(C_3^{(2)}, C_4^{(1)})$, then it indicates that $C^{(1)}$ and $C^{(2)}$ contain the same keyword. They wrote that:

> (C: Efficiency Optimization [8]): *Via linking ciphertexts embedded with the same keyword together, once a ciphertext is matched with the queried trapdoor, all the matched ciphertexts sent from the sender* S *will be directly returned. The efficiency optimization method is similar to that of PAUKS [7].*

From this statement, two points become clear: (1) C_3 and C_4 must belong to different groups, i.e., $\bar{C}_3 = C_3 \in \mathbb{G}_1$ and $\bar{C}_4 = C_4 \in \mathbb{G}_2$, and (2) the checking equation $e(C_3^{(1)}, C_4^{(2)}) = e(C_3^{(2)}, C_4^{(1)})$ can be employed to break the security, as in our attack: $\mathcal{A}$ obtains an updated ciphertext for w_0^*, that contains $(\bar{C}_3, \bar{C}_4) = (C_3, C_4)$, and the challenge ciphertext, and checks the equation holds or not. That is, even if our attack, that employed $(\bar{C}_1, \bar{C}_2) = (C_1, C_2)$, is protected via asymmetric pairings, still keyword information is leaked from $(\bar{C}_3, \bar{C}_4) = (C_3, C_4)$. We conclude that the simple correction (i.e., replacing symmetric pairings with asymmetric pairings to obtain DDH-hard groups) is not sufficient to prevent our attack. Furthermore, enabling linkability to improve search efficiency inevitably leads to leakage of keyword-related information from ciphertexts. As discussed above, the ability to link ciphertexts without performing a search reveals that the same keyword appears in multiple documents. This behavior can hardly be regarded as appropriate or desirable functionality for a searchable encryption scheme. If further modifications are to be considered, one possible direction would be to prioritize security by eliminating linkability altogether, for example by mapping all ciphertexts to the same group elements to prevent the DDH test. However, we emphasize that this approach only prevents our specific attack employing the DDH test and does not constitute a complete security proof. As a next step, it is necessary to introduce appropriate complexity assumptions and formally prove security.

6 Conclusion

In this short note, we demonstrate that keyword information can be leaked from ciphertexts in the proxy-free PAUKS scheme [8], and mention that our attack is effective against the PAUKS scheme [7]. Furthermore, even if our attack can be prevented by weakening the security model or modifying the oracle inputs, we argue that a realistic threat remains. To modify the scheme, it is

first necessary to refine the security model. In particular, it should be clarified whether an update ciphertext query requires a ciphertext as input. Only after refining the security model to properly capture the intended functionality can a secure scheme be meaningfully proposed. Addressing these issues is left as future work and is beyond the scope of this short note.

Acknowledgment: This work was supported by JSPS KAKENHI Grant Number JP26K02908.

References

[1] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[2] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[3] Leixiao Cheng and Fei Meng. Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor. *IEEE Transactions on Information Forensics and Security*, 19:1388–1400, 2024.

[4] Keita Emura. Comments on "Lightweight multi-user public-key authenticated encryption with keyword search". *IEEE Transactions on Information Forensics and Security*, 21:5108–5109, 2026.

[5] Keita Emura. Security analysis on the multi-user keyword-field-free conjunctive searchable encryption scheme. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E109.A(8):1419–1420, 2026.

[6] Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14, 2017.

[7] Hongbo Li, Qiong Huang, Jianye Huang, and Willy Susilo. Public-key authenticated encryption with keyword search supporting constant trapdoor generation and fast search. *IEEE Transactions on Information Forensics and Security*, 18:396–410, 2023.

[8] Hongbo Li, Willy Susilo, Jian Shen, Chen Wang, Leixiao Cheng, and Qiong Huang. Proxy-free public-key authenticated updatable and searchable encryption for cloud storage. *IEEE Transactions on Dependable and Secure Computing*, 23(1):236–249, 2026.

[9] Qinyi Li and Xavier Boyen. Public-key authenticated encryption with keyword search made easy. *IACR Communications in Cryptology*, 1(2):16, 2024.

[10] Tatsuaki Okamoto and David Pointcheval. The Gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, pages 104–118, 2001.

[11] Wei Wang, Dongli Liu, Zilin Zheng, Peng Xu, and Laurence Tianruo Yang. Identity-based group encryption with keyword search against keyword guessing attack. *IEEE Transactions on Information Forensics and Security*, 19:8023–8036, 2024.

[12] Yongliang Xu, Hang Cheng, Jiguo Li, Ximeng Liu, Xinpeng Zhang, and Meiqing Wang. Lightweight multi-user public-key authenticated encryption with keyword search. *IEEE Transactions on Information Forensics and Security*, 20:3234–3246, 2025.

[13] Wenjie Yang, Shujie Lin, Futai Zhang, and Anjia Yang. Dual-server public-key authenticated searchable encryption with constant trapdoor against inside keyword guessing attacks. *IEEE Transactions on Information Forensics and Security*, 21:5794–5807, 2026.

[14] Takeshi Yoshida and Keita Emura. Comments on "Identity-based group encryption with keyword search against keyword guessing attack"'. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E109.A(7):1314–1315, 2026.

[15] Takeshi Yoshida and Keita Emura. Comments on "Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor". Cryptology ePrint Archive, Paper 2026/1015, 2026. `https://eprint.iacr.org/2026/1015`.

[16] Shengjie Zhang, Jianchang Lai, Liquan Chen, Jinguang Han, and Ge Wu. Keyword-field-free conjunctive searchable encryption for multiuser in EMR system. *IEEE Internet of Things Journal*, 12(15):30850–30861, 2025.