Security Analysis on a Public-Key Inverted-Index Keyword Search Scheme with Designated Tester

Mizuki Hayashi[§] and Keita Emura[§,†]

[§]Kanazawa University, Japan.
[†]AIST, Japan.

June 19, 2025

Abstract

Gao et al. (IEEE Internet of Things Journal 2024) proposed public-key inverted-index keyword search with designated tester as an extension of public key encryption with keyword search (PEKS). In their scheme, a server (a tester) has a secret key and uses the key for running the search algorithm due to the designated tester setting. They proved that no information of keyword is revealed from trapdoors under the decisional Diffie-Hellman (DDH) assumption. However, they also employed a symmetric pairing which can be seen as a DDH-solver. Thus, it is expected that information of keyword is revealed from trapdoors since the underlying complexity assumption does not hold. In this paper, we demonstrate two attacks against the Gao et al.'s scheme where information of keyword is revealed from a trapdoor. The first attack completes by using only the server's secret key in addition to the challenge trapdoor, without any additional encryption/trapdoor queries. We remark that an adversary is not allowed to obtain the server's secret key in their security model, and our attack is outside of their security model. Thus, we discuss the roles of the server, and stress that our attack scenario is reasonable. The second attack does not employ the server's secret key and utilizes linkability of two trapdoors. In both attacks, the attack complexity is just two pairing computations and is feasible in terms of the computational cost.

1 Introduction

Public key encryption with keyword search (PEKS) was proposed by Boneh et al. [5]. In PEKS, a sender encrypts a keyword w using the receiver's public key, and the receiver generates a trapdoor associated with a keyword to be searched w' using the receiver's secret key. The test algorithm, that takes a ciphertext and a trapdoor, checks whether the encrypted keyword and the searched keyword are the same or not. In addition to correctness (the test algorithm outputs 1 if $w = w'$) and consistency (the test algorithm outputs 0 if $w \neq w'$), it is required that no information of keyword is revealed from ciphertexts (IND-CKA: Indistinguishability against chosen keyword attacks). In the definition of IND-CKA, an adversary declares two challenge keywords, and it guarantees that the adversary cannot distinguish which challenge keyword is encrypted. Here, the adversary is allowed to obtain trapdoors for non-challenge keywords. See [1] for the definitions in detail.

The following keyword guessing attack is widely recognized as a well-known issue in PEKS. Suppose an adversary possesses a trapdoor and intends to extract information about the keyword associated with it. The adversary selects a keyword, encrypts it, and executes the test algorithm

using the ciphertext and the trapdoor. Based on the results, the adversary can determine whether the chosen keyword is associated with the trapdoor. To resist the keyword guessing attack, mainly two primitives have been proposed: PAEKS (public key authenticated encryption with keyword search) [13, 15] where a sender's secret key is required to encrypt a keyword, and designated-tester PEKS [2, 17] where the server's (tester's) secret key is required to run the test algorithm.

Gao et al. [10] proposed public-key inverted-index keyword search with designated tester. To run the test algorithm the server's secret key is required due to the designated-tester setting. Moreover, as in PAKES, a sender's secret key is required to encrypt a keyword.

Our Motivation. Gao et al. proved that no information of keyword is revealed from trapdoors under the decisional Diffie-Hellman (DDH) assumption, i.e., for $(g, g^a, g^b, g^c) \in \mathbb{G}^4$ (here $\mathbb{G}$ be a group), it is assumed to be infeasible to decide $c = ab$ or not. However, they also employed a symmetric pairing which can be seen as a DDH-solver [16]. Concretely, let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing (here $\mathbb{G}_T$ be a group which is so-called a target group). Then, for an DDH instance over the source group $\mathbb{G}$, i.e., $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, one can efficiently check whether $c = ab$ or not by checking whether $e(g^a, g^b) = e(g^c, g)$ holds or not. Thus, the Gao et al.'s scheme is not provably secure since the underlying complexity assumption does not hold. Note that this situation does not immediately break the Gao et al.'s scheme. Thus, it would be better to confirm whether information of keyword is actually revealed from a trapdoor or not.

Our Contribution. In this paper, we demonstrate two attacks against the Gao et al.'s scheme where information of keyword is revealed from a trapdoor. The first attack completes by using the challenge trapdoor and the server's secret key only, without any additional encryption/trapdoor queries, and the attack complexity is just two pairing computations. We remark that in their security model an adversary is not allowed to obtain the server's secret key and our attack is outside of their security model. Thus, we examine whether our attack scenario is reasonable or not by considering the roles of the server. The second attack utilizes linkability of two trapdoors. We show that information of keyword is revealed from a trapdoor without using the server's secret key if an adversary is allowed to obtain a trapdoor of the challenge keyword as in the security definition of PAEKS which is so called full trapdoor indistinguishabilty (TI) [15].

Related Work. Zhang et al. [18] pointed out a vulnerability of the Guo et al.'s designated-server PAEKS scheme [11]. They also focused on the fact that the DDH assumption does not hold under the symmetric pairing setting, as in ours. Moreover, their attack also uses the server's secret key, as in ours. However, the Gao et al.'s scheme (which is the target scheme in this paper) and the Guo et al.'s scheme are constructed under a different methodology, and thus the Zhang et al.'s attack is totally different from ours. For example, the Zhang et al.'s attack needs to issue an encryption query to obtain a ciphertext whereas no encryption query is required in our attack. This difference explains a difference of the attack strategy between the Zhang et al.'s attack and our attack well.

Emura [9] pointed out a vulnerability of the Jiang et al.'s forward secure PAEKS scheme [14]. He also focused on the fact that the DDH assumption does not hold under the symmetric pairing setting, as in ours. However, his attack strategy is different from ours because his attack requires that an adversary issues encryption and trapdoor queries unlike our attack.

Differences from the proceedings version. An extended abstract appeared at the 20th Asia Joint Conference on Information Security (AsiaJCIS) 2025 [12] that introduced the first attack. In this full version, we newly propose the second attack where information of keyword is revealed from a trapdoor without using the server's secret key under the full TI setting. We focus on the fact that whether two plaintexts of two ElGamal ciphertexts are the same or not can be checked via a pairing computation. We summarize our two attacks in Table 1.

	Trapdoor Query	Encryption Query	The Server's Secret key $\mathsf{sk}_{\mathsf{IS}}$
First Attack (Sec. 4.1)	No	No	Yes
Second Attack (Sec. 4.2)	Yes (for the challenge keyword)	No	No

Table 1: Summary of Our Attacks

2 Security Models given by Gao et al.

In this section, we introduce the security model given by Gao et al. [10]. They define a cloud server and a proxy server, in addition to a sender, a receiver, and the server (which is called index server in their paper). Moreover, they consider a decryption procedure in addition to the search functionality. In this paper, we focus on only the search functionality and omit other functionalities. Remark that Gao et al. called the Enc algorithm Index Generation, the $\mathsf{Trapdoor}$ algorithm Trapdoor Generation, and the Test algorithm Index Server Retrieval. We employ the notations according to those of PEKS/PAEKS.

- Setup: The setup algorithm takes a security parameter $\lambda \in \mathbb{N}$ as input, and outputs a common parameter params.
- $\mathsf{KeyGen}_{\mathsf{S}}$: The sender's key generation algorithm takes params as input, and outputs a pair of a sender public key and secret key $(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$.
- $\mathsf{KeyGen}_{\mathsf{R}}$: The receiver's key generation algorithm takes params as input, and outputs a pair of a receiver public key and secret key $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}})$.
- $\mathsf{KeyGen}_{\mathsf{IS}}$: The server's (tester's) key generation algorithm takes params as input, and outputs a pair of a server's public key and secret key $(\mathsf{pk}_{\mathsf{IS}}, \mathsf{sk}_{\mathsf{IS}})$.
- Enc: The keyword encryption algorithm takes sk_{S}, pk_{R}, $\mathsf{pk}_{\mathsf{IS}}$, and a keyword w as input, and outputs a ciphertext ct.
- $\mathsf{Trapdoor}$: The trapdoor generation algorithm takes sk_{R}, $\mathsf{pk}_{\mathsf{IS}}$, and a keyword w' as input, and outputs a trapdoor td.
- Test: The test algorithm takes ct, td, and $\mathsf{sk}_{\mathsf{IS}}$ as input, and outputs 1 or 0.

In PAEKS [13, 15], a trapdoor is generated by indicating a sender public key (i.e., the $\mathsf{Trapdoor}$ algorithm takes pk_{S} as input in addition to sk_{R}), and thus a trapdoor depends on a sender. On the other hand, the $\mathsf{Trapdoor}$ algorithm does not take pk_{S} as input in the syntax given by Gao et al.

Gao et al. defined IND-CKA (indistingusiability of keyword against chosen keyword attack) and IND-KGA (indistingusiability of keyword against keyword guessing attack). IND-CKA that guarantees that no information of keyword is revealed from ciphertexts, and IND-GKA guarantees that no information of keyword is revealed from trapdoors, respectively. An adversary declares two challenge keywords, (w_0^*, w_1^*). For $b \xleftarrow{\$} \{0,1\}$, the adversary and obtains a ciphertext of w_b^* (the challenge ciphertext) in IND-CKA or a trapdoor of w_b^* (the challenge trapdoor) in IND-KGA, respectively. In these security models, the adversary is allowed to obtain public keys (pk_{S}, pk_{R}, $\mathsf{pk}_{\mathsf{IS}}$), namely, the server that runs the Test algorithm is not regarded as an adversary. Moreover, the adversary is allowed to obtain trapdoors of non-challenge keyword, and is not allowed to obtain any ciphertext. These security models are quite weak and do not capture real situations. For example, no security is guaranteed after an adversary observes a ciphertext even just once.

Roles of the Server (Tester). Here, we re-consider the roles of the server. The basic flow is as follows. A sender sends a ciphertext ct to the server and a receiver sends a trapdoor td to the server. The server runs the Test algorithm and executes a procedure depending on the test result (in the Gao et al.'s scheme, the server returns a ciphertext of a document containing keywords to the receiver). Since the server is allowed to search a keyword by running the Test algorithm, the server can decide whether an encrypted keyword is the same as a keyword to be searched when the server obtains ct and td. More precisely, the information is intentionally revealed from ct and td to enable the search functionality over an encrypted form.

Security against the Server. First of all, we emphasize that any searchable encryption is not necessary if the server is trusted. For example, if the server is trusted, then the following simple protocol enables a secure keyword search. The server manages a public key and a decryption key of a PKE scheme. A sender encrypts a keyword using the server's public key and sends the ciphertext to the server. A receiver also encrypts a keyword to be searched using the server's public key and sends the ciphertext to the server as a trapdoor. Then, due to the security of PKE, no eavesdropper can obtain information of keyword from the ciphertext and the trapdoor. The server decrypts both the ciphertext and the trapdoor using own decryption key, and checks whether these keywords are the same or not.

The fact that the server performs searches for encrypted keywords suggests the following motivation:

> *No information about the keywords "beyond enabling the search functionality" should be revealed to the server.*

Thus, a security model where an adversary is modeled as the server is indispensable. Since the server is not allowed to produce a ciphertext (unlike PEKS), there is a room for defining a keyword guessing attack resistance against the server where the server does not obtain information of keyword from trapdoors (beyond enabling the search functionality) as follows. Here, an adversary needs to be allowed to obtain the server's secret key $\mathsf{sk}_{\mathsf{IS}}$ in addition to public keys (pk_{S}, pk_{R}, $\mathsf{pk}_{\mathsf{IS}}$). The adversary declares two challenge keywords (w_0^*, w_1^*) and obtains the challenge trapdoor that is a trapdoor for w_b^* where $b \xleftarrow{\$} \{0,1\}$ and wins if the adversary correctly guesses b.[1] Since the adversary is modeled as the server who is sent a ciphertext from the sender, the adversary should be allowed to issue an encryption query where the adversary queries a keyword $w \notin \{w_0^*, w_1^*\}$, and obtains the ciphertext. Here, we need to prohibit that the adversary sends $w \in \{w_0^*, w_1^*\}$ as an encryption query since it enables the adversary to run the Test algorithm with the challenge trapdoor that trivially distinguishes b. On the other hand, there is no trivial attack if the adversary is allowed to obtain a trapdoor for $w \in \{w_0^*, w_1^*\}$ (in addition to the challenge trapdoor). This security model is formalized as full trapdoor indistingishiability (full TI) in PAEKS [15] that requires the unlinkability of trapdoors. If the adversary is not allowed to obtain a trapdoor for $w \in \{w_0^*, w_1^*\}$, then the security model is formalized as TI. To prevent the keyword guessing attack against the server, at least TI security is necessary. We also remark that security against the server implies security against a third entity who observes public keys (pk_{S}, pk_{R}, $\mathsf{pk}_{\mathsf{IS}}$) and trapdoors/ciphertexts.

3 Gao et al.'s Scheme

In this section, we introduce the Gao et al.'s scheme. As mentioned above, we focus on the search functionality and omit other functionalities. Though Gao et al. employ the inverted index in their

[1] The advantage of the adversary $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ where b' is the output of $\mathcal{A}$.

scheme, we also omit it since it does not depend on our attack. Though the Enc algorithm does not take $\mathsf{pk_S}$ as input in their syntax, we explicitly add $\mathsf{pk_S}$ as input because V is used in the Enc algorithm where $\mathsf{pk_S} = (V, g^{\mathsf{sk_S}})$. On the contrary, $\mathsf{sk_S}$ is not explicitly used in the following because it is independent to the search functionality.

Gao et al.'s Scheme (Simplified)

- $\mathsf{Setup}(1^\lambda)$: Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups with prime order $p > 2^\lambda$, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a pairing, and $g \in \mathbb{G}$ be a generator. Let $H_1 : \mathbb{G}_T \to \mathbb{Z}_p^*$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ be collision resistant hash functions. Output $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, e, g, H_1, H_2)$.
- $\mathsf{KeyGen_S}(\mathsf{params})$: Choose $\mathsf{sk_S} \xleftarrow{\$} \mathbb{Z}_p^*$ and $V \xleftarrow{\$} \mathbb{G}$, set $\mathsf{pk_S} = (V, g_{\mathsf{S}}^{\mathsf{sk}})$, and output $(\mathsf{pk_S}, \mathsf{sk_S})$.
- $\mathsf{KeyGen_R}(\mathsf{params})$: Choose $\mathsf{sk_R} \xleftarrow{\$} \mathbb{Z}_p^*$, compute $\mathsf{pk_R} = g^{\mathsf{sk_R}}$, and output $(\mathsf{pk_R}, \mathsf{sk_R})$.
- $\mathsf{KeyGen_{IS}}(\mathsf{params})$: Choose $\mathsf{sk_{IS}} \xleftarrow{\$} \mathbb{Z}_p^*$, compute $\mathsf{pk_{IS}} = g^{\mathsf{sk_{IS}}}$, and output $(\mathsf{pk_{IS}}, \mathsf{sk_{IS}})$.
- $\mathsf{Enc}(\mathsf{sk_S}, \mathsf{pk_S}, \mathsf{pk_R}, \mathsf{pk_{IS}}, w)$: Choose $r, s \xleftarrow{\$} \mathbb{Z}_p^*$, compute
  - $C_{w,1} = (\mathsf{pk_R})^r$,
  - $t = e(\mathsf{pk_{IS}}, V)^s$,
  - $C_{w,2} = H_1(e(g, g^{H_2(w)})^r \cdot t)$, and
  - $C_{w,3} = g^s$,

  and outputs $\mathsf{ct} = (C_{w,1}, C_{w,2}, C_{w,3})$.
- $\mathsf{Trapdoor}(\mathsf{sk_R}, \mathsf{pk_{IS}}, w')$: Choose $\zeta \xleftarrow{\$} \mathbb{Z}_p^*$, compute
  - $T_{w',1} = (g^{H_2(w')})^{1/\mathsf{sk_R}} \cdot \mathsf{pk_{IS}}^{\zeta}$ and
  - $T_{w',2} = g^{\zeta}$,

  and output $\mathsf{td} = (T_{w',1}, T_{w',2})$.
- $\mathsf{Test}(\mathsf{ct}, \mathsf{td}, \mathsf{sk_{IS}})$: Parse $\mathsf{ct} = (C_{w,1}, C_{w,2}, C_{w,3})$ and $\mathsf{td} = (T_{w',1}, T_{w',2})$. Compute
  - $t' = e(C_{w,3}, V)^{\mathsf{sk_{IS}}}$ and
  - $R = H_1(e(C_{w,1}, T_{w',1}/T_{w',2}^{\mathsf{sk_{IS}}}) \cdot t)$.

  Output 1 if $R = C_{w,2}$ and 0, otherwise.

If a ciphertext ct and a trapdoor td are generated according to the algorithm description, then the following holds.

$$
\begin{aligned}
e(C_{w,1}, T_{w',1}/T_{w',2}^{\mathsf{sk_{IS}}}) \cdot t' &= e((\mathsf{pk_R})^r, (g^{H_2(w')})^{1/\mathsf{sk_R}} \cdot \mathsf{pk_{IS}}^{\zeta}/(g^{\zeta})^{\mathsf{sk_{IS}}}) \cdot e(g^s, V)^{\mathsf{sk_{IS}}} \\
&= e((\mathsf{pk_R})^r, (g^{H_2(w')})^{1/\mathsf{sk_R}} \cdot \mathsf{pk_{IS}}^{\zeta}/(g^{\mathsf{sk_{IS}}})^{\zeta}) \cdot e(g^{\mathsf{sk_{IS}}}, V)^s \\
&= e((\mathsf{pk_R})^r, (g^{H_2(w')})^{1/\mathsf{sk_R}} \cdot \mathsf{pk_{IS}}^{\zeta}/\mathsf{pk_{IS}}^{\zeta}) \cdot e(\mathsf{pk_{IS}}, V)^s \\
&= e((\mathsf{pk_R})^{1/\mathsf{sk_R}}, g^{H_2(w')})^r \cdot t \\
&= e(g, g^{H_2(w')})^r \cdot t
\end{aligned}
$$

where $t = e(\mathsf{pk}_{\mathsf{IS}}, V)^s$ is the value to compute ct. Thus, $R = C_{w,2}$ holds if $w = w'$.

Construction Methodology of the Gao et al.'s scheme. Before giving our attacks, we explore the reason behind our attacks work by exploring the construction methodology of the Gao et al.'s scheme. We observe that a trapdoor

- $T_{w',1} = (g^{H_2(w')})^{1/\mathsf{sk}_{\mathsf{R}}} \cdot \mathsf{pk}_{\mathsf{IS}}^{\zeta}$
- $T_{w',2} = g^{\zeta}$

can be regarded as an ElGamal ciphertext [7] on the (keyword-related) plaintext $((g^{H_2(w')})^{1/\mathsf{sk}_{\mathsf{R}}})$ under the public key $\mathsf{pk}_{\mathsf{IS}}$ and the randomness ζ. The following procedure in the Test algorithm corresponds to the ElGamal decryption algorithm.

- $T_{w',1}/T_{w',2}^{\mathsf{sk}_{\mathsf{IS}}} = (g^{H_2(w')})^{1/\mathsf{sk}_{\mathsf{R}}}$

By the pairing computation with pk_{R}, $1/\mathsf{sk}_{\mathsf{R}}$ is canceled out and $g^{H_2(w')}$ is obtained.

Behind the First Attack. The procedure in the first attack

$$e(T_{w_b^*,1}/T_{w_b^*,2}^{x}, \mathsf{pk}_{\mathsf{R}})$$

extracts the above "ElGamal decryption" part from the Test algorithm. The keyword-related plaintext $((g^{H_2(w')})^{1/\mathsf{sk}_{\mathsf{R}}})$ is obtained and thus we can extract information of keyword.

Behind the Second Attack. The second attack employs the fact that a symmetric pairing allows us to check whether two plaintexts of two ElGamal ciphertexts are the same or not. That is, for two trapdoors $(T_{w,1}, T_{w,2})$ and $(T_{w',1}, T_{w',2})$,

$$e(T_{w,1}/T_{w',1}, g) = e(\mathsf{pk}_{\mathsf{IS}}, T_{w,2}/T_{w',2})$$

holds if $w = w'$. The second attack utilizes this linkability of two trapdoors.

4 Proposed Attacks

In this section, we give two attacks.

4.1 First Attack with the Server's Secret Key

As mentioned above, an adversary is modeled as a malicious server and obtains $(\mathsf{pk}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{IS}}, \mathsf{sk}_{\mathsf{IS}})$. Our attack model is weak in the sense that neither trapdoor query nor encryption query is allowed. The adversary declares two challenge keywords (w_0^*, w_1^*). The challenger randomly chooses $b \xleftarrow{\$} \{0,1\}$, computes the challenge trapdoor $\mathsf{td}^* \leftarrow \mathsf{Trapdoor}(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{IS}}, w_b^*)$, and gives $\mathsf{td}^* = (T_{w_b^*,1}, T_{w_b^*,2})$ to the adversary.

We show that the adversary correctly decides b as follows. Here, the following equations hold.

$$e(T_{w_b^*,1}, \mathsf{pk}_{\mathsf{R}}) = e(g^{H_2(w_b^*)}, g)e(X^{\zeta}, \mathsf{pk}_{\mathsf{R}}) = e(g^{H_2(w_b^*)}, g)e(T_{w_b^*,2}^{x}, \mathsf{pk}_{\mathsf{R}})$$

Then,

$$e(T_{w_b^*,1}/T_{w_b^*,2}^{x}, \mathsf{pk}_{\mathsf{R}}) = e(g^{H_2(w_b^*)}, g)$$

holds. The adversary decides b by using $\mathsf{td}^* = (T_{w_b^*,1}, T_{w_b^*,2})$ and $\mathsf{sk}_{\mathsf{IS}} = x$ as follows.

$$e(T_{w_b^*,1}/T_{w_b^*,2}^{x}, \mathsf{pk}_{\mathsf{R}}) = e(g^{H_2(w_0^*)}, g)$$

If the above equation holds, then the adversary outputs $b = 0$. Note that td^* is a trapdoor of either w_0^* or w_1^* due to the security model. Thus, the adversary outputs $b = 1$ if the above equation does not hold. In summary, the adversary correctly decides b with probability 1.

4.2 Second Attack without the Server's Secret Key

In this attack, an adversary is modeled as a third entity that obtains public keys $(\mathsf{pk_S}, \mathsf{pk_R}, \mathsf{pk_{IS}})$. In addition to these keys, the adversary is allowed to issue a trapdoor query for the challenge keyword. Note that the adversary does not have to issue other trapdoor queries and encryption queries. We also remark that the security model is called full TI (Trapdoor Indistinguishability) in PAEKS [15]. Thus, the second attack shows that the Gao et al.'s scheme does not provide the full TI security.

The adversary declares two challenge keywords (w_0^*, w_1^*). The challenger randomly chooses $b \xleftarrow{\$} \{0,1\}$, computes the challenge trapdoor $\mathsf{td}^* \leftarrow \mathsf{Trapdoor}(\mathsf{sk_R}, \mathsf{pk_{IS}}, w_b^*)$, and gives $\mathsf{td}^* = (T_{w_b^*,1}, T_{w_b^*,2})$ to the adversary. The adversary issues a trapdoor query w_0^* and obtains $(T_{w_0^*,1}, T_{w_0^*,2})$. Here, $T_{w_0^*,1}/T_{w_b^*,1} = (g^{H_2(w_0^*)})^{1/\mathsf{sk_R}} \cdot \mathsf{pk_{IS}}^{\zeta} / (g^{H_2(w_b^*)})^{1/\mathsf{sk_R}} \cdot \mathsf{pk_{IS}}^{\zeta^*}$ where ζ and ζ^* are randomness for $T_{w_0^*,1}$ and $T_{w_b^*,1}$, respectively. If $b = 0$, then

$$T_{w_0^*,1}/T_{w_b^*,1} = \mathsf{pk_{IS}}^{\zeta-\zeta^*}$$

holds. Moreover,

$$T_{w_0^*,2}/T_{w_b^*,2} = g^{\zeta-\zeta^*}$$

holds. Thus, the adversary decides $b = 0$ if

$$e(T_{w_0^*,1}/T_{w_b^*,1}, g) = e(\mathsf{pk_{IS}}, T_{w_0^*,2}/T_{w_b^*,2})$$

holds. On the other hand, if $b = 1$, then

$$e(T_{w_0^*,1}/T_{w_b^*,1}, g) \neq e(\mathsf{pk_{IS}}, T_{w_0^*,2}/T_{w_b^*,2})$$

holds since $e(g,g)^{(H_2(w_0^*)-H_2(w_1^*))/\mathsf{sk_R}} \neq 1$ due to non-degeneracy: $e(g,g) \neq 1$ and collision resistance of H_2: for $w_0^* \neq w_1^*$, $H_2(w_0^*) \neq H_2(w_1^*)$ with overwhelming probability. Thus, the adversary correctly decides b with probability 1 minus the probability that the collision happens.

5 Attack Complexity

Our attacks are completed with two pairing computations. Here, we estimate the actual computation time of our attack when a symmetric pairing is employed. In general, an asymmetric pairing is effective for efficiently implementing cryptographic schemes. Thus, pairing libraries (e,g, mcl[2]) usually implement only asymmetric pairings. As an exception, the PBC library[3] implements symmetric pairings. In the library, two types are implemented, Type A and Type A1. In a type A curve, the size of the finite field ($\mathbb{G}_T$) is 1024 bits, and it is not sufficient to assume the hardness of the discrete logarithm problem over $\mathbb{G}_T$. In general, the size is required to be 3072 bits for realizing 128-bit security.[4] However, no such a curve is implemented in PBC. Therefore, we employ a Type A1 curve whose finite field size is 2048 bits, that realizes 80-bit security. The benchmark of a pairing computation over a type A1 curve (given in the PBC web page) is 757 milliseconds, and the computation time of two pairings is roughly estimated to be less than two seconds. Therefore, we conclude that our attack completes less than two seconds and is reasonable from the viewpoint of computational complexity.[5]

[2]mcl: a portable and fast pairing-based cryptography library. Available at `https://github.com/herumi/mcl`.

[3]PBC: Pairing-Based Cryptography library (Ver. 0.5.14). Available at `https://crypto.stanford.edu/pbc/`.

[4]NIST SP 800-57 Part 1 Rev. 5 Recommendation for Key Management: Part 1 - General (`https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf`).

[5]Ema et al. [8] generated a parameter with 128-bit security using `PairingParametersGenerator` API [6]. In their implementation result, one pairing computation is less than 100 milliseconds. The difference between the benchmark given in the PBC web page and that of Ema et al.'s seem to be caused by the machine spec. In this paper, we follow the original benchmark to clarify still our attack is reasonable even somewhat old machine is employed.

6 Possibility of Correction

In this section, we explore whether the Gao et al.'s scheme can be corrected to protect our attacks. Note that the ElGamal scheme is IND-CPA secure under the DDH assumption (where IND-CPA stands for Indistinguishability against chosen plaintext attacks) that guarantees no information of plaintext is revealed from ciphertexts. In summary, it is natural to leak information about keyword from trapdoors under the symmetric pairing setting since the DDH problem is not hard.

Gao et al.'s Scheme over Asymmetric Pairings. As a natural motivation, it should be clarified whether our attacks still work or not under asymmetric pairings [3, 4] that the DDH assumption holds.

The first attack still works even under asymmetric pairings since no security of ElGamal contributes to hide information of the keyword-related plaintext against an adversary who has the decryption key $\mathsf{sk}_{\mathsf{IS}} = x$. We concretely confirm it as follows. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups with prime order p, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric pairing (that is, it is assumed that no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exists). Then, the DDH assumption holds over both $\mathbb{G}_1$ and $\mathbb{G}_2$, and it is called the SXDH (Symmetric eXternal Diffie-Hellman) assumption. Next, we assign each element to either $\mathbb{G}_1$ or $\mathbb{G}_2$ as follows. A trapdoor and pk_{R} belong to different group due to the pairing computation. Thus, we set $\mathsf{pk}_{\mathsf{R}} \in \mathbb{G}_1$ and $T_{w',1}, T_{w',2} \in \mathbb{G}_2$ without loss of generality. Then, from the form of $T_{w',1}$, $\mathsf{pk}_{\mathsf{IS}} \in \mathbb{G}_2$ holds. Moreover, $V \in \mathbb{G}_1$ due to the form of t. Under this setting, a ciphertext is re-described as follows.

- $C_{w,1} = (\mathsf{pk}_{\mathsf{R}})^r \in \mathbb{G}_1$
- $t = e(V, \mathsf{pk}_{\mathsf{IS}})^s \in \mathbb{G}_T$
- $C_{w,2} = H_1(e(g_1, g_2^{H_2(w)})^r \cdot t)$ (here $t = e(V, \mathsf{pk}_{\mathsf{IS}})^s$)
- $C_{w,3} = g_2^s \in \mathbb{G}_2$

Since $t' = e(V, C_{w,3})^{\mathsf{sk}_{\mathsf{IS}}}$, $H_1(e(C_{w,1}, T_{w',1}/T_{w',2}^{\mathsf{sk}_{\mathsf{IS}}}) \cdot t') = C_{w,2}$ holds if $w = w'$. That is, replacing a symmetric pairing to an asymmetric pairing does not affect the functionality of the Test algorithm of the Gao et al.'s scheme. As mentioned above, the first attack employs a part of the Test algorithm. Thus, the first attack still works as follows.

- $e(\mathsf{pk}_{\mathsf{R}}, T_{w_b^*,1}/T_{w_b^*,2}^x) = e(g_1^{H_2(w_0^*)}, g_2)$

If the above equation holds, then the adversary outputs $b = 0$, and $b = 1$ otherwise. Thus, replacing a symmetric pairing to an asymmetric pairing does not prevent the first attack.

On the other hand, the second attack does not work well since it totally depends on the functionality of the symmetric pairing. Concretely, since $T_{w',2}, \mathsf{pk}_{\mathsf{IS}} \in \mathbb{G}_2$, we cannot compute $e(\mathsf{pk}_{\mathsf{IS}}, T_{w_0^*,2}/T_{w_b^*,2})$. Note that this fact does not formally prove that the Gao et al.'s scheme over asymmetric pairings provides the full TI security.

Finally, we reconsider the fact that the $\mathsf{Trapdoor}$ algorithm in the Gao et al.'s scheme does not take the sender's public key pk_{S} as input whereas that of PAEKS takes pk_{S} as input. Moreover, the Enc algorithm in the Gao et al.'s scheme does not explicitly use sk_{S} whereas that of PAEKS uses sk_{S}. Introducing the sender's key in the $\mathsf{Trapdoor}$ and Enc algorithms may be effective to prevent an attack by the adversary who has $\mathsf{sk}_{\mathsf{IS}}$. Here, remember that their security models are quite weak and do not capture real situations. For example, no security is guaranteed after an adversary observes a ciphertext even just once. That is, even if the first attack will be protected by adding the sender's key in the $\mathsf{TrapdoorEnc}$ algorithms, we need to modify not only the syntax

but also the security models. This situation is the same as proposing a new scheme. We conclude that correcting the Gao et al.'s scheme is non-trivial due to the above reasons.

7 Relevance of Our Attack Models

In the first attack, we consider a situation where the adversary is given only $\mathsf{sk}_{\mathsf{IS}}$ in addition to the challenge trapdoor. Note that the adversary is not allow to obtain any other trapdoor or ciphertext. Considering the role of the server (tester), the adversary should be allowed to obtain additional trapdoor and ciphertext. In this perspective, our attack model is much weaker than the one described above. As mentioned above, any searchable encryption is not necessary if the server is trusted. Thus, a security model where an adversary is modeled as the server is indispensable. Therefore, we conclude that our setting that the server is an adversary is reasonable. Here, an adversary modeled as the server needs to be allowed to obtain the server's secret key $\mathsf{sk}_{\mathsf{IS}}$ in addition to public keys (pk_{S}, pk_{R}, $\mathsf{pk}_{\mathsf{IS}}$).

In the second attack, the adversary needs to be allowed to obtain a trapdoor of the challenge keyword in addition to the challenge trapdoor. Since the security model is weak in the sense that the adversary does not have to issue an encryption query, and the full TI security has been introduced in the PAEKS context, we conclude that our setting is reasonable. Note that the full TI security has been recognized as a stronger model in PAEKS, and the TI security, where an adversary is allowed to issue trapdoor queries for non-challenge keywords, is also meaningful (See [15]). In this perspective, we conclude that the security model of the first attack is more reasonable than that of the second attack even though the adversary has $\mathsf{sk}_{\mathsf{IS}}$ since the server should be considered as an adversary as discussed above.

Gao et al. stated that the server follows the protocol description. Precisely, they stated that "*In the system, we consider that the index server executes the index and trapdoor retrieval correctly. If the retrieval is success, it sends the legitimate receiver identity to proxy server and inverted list to receiver.*". We confirm that the adversaries of our attacks do not deviate the procedure of the protocol because they complete the proposed two attacks locally. We conclude that our attack models are reasonable because we do not violate their statement above and the above mentioned reasons.

8 Conclusion

In this paper, we show that information of keyword is leaked from trapdoors in the Gao et al.'s scheme. We also show that the computational cost of the attacks is about two seconds and replacing a symmetric pairing to an asymmetric pairing does not prevent the first attack. We also discussed relevance of our attack models and conclude that the models are reasonable.

Acknowledgment: This work was partially supported by JSPS KAKENHI Grant Number JP21K11897.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, 2008.

[2] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *ICCSA*, pages 1249–1259, 2008.

[3] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *SCN*, pages 257–267, 2002.

[4] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *SAC*, pages 319–331, 2005.

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

[6] Angelo De Caro and Vincenzo Iovino. jPBC: Java pairing based cryptography. In *ISCC*, pages 850–855. IEEE, 2011.

[7] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[8] Shuntaro Ema, Yuta Sato, Hinata Nishino, Keita Emura, and Toshihiro Ohigashi. Implementation and evaluation of an identity-based encryption with security against the KGC. *Journal of Information Processing*, 33:185–196, 2025.

[9] Keita Emura. Generic construction of forward secure public key authenticated encryption with keyword search. In *Applied Cryptography and Network Security*, pages 237–256, 2024.

[10] Nan Gao, Kai Fan, Haoyang Wang, Kuan Zhang, Hui Li, and Yintang Yang. Public-key inverted-index keyword search with designated tester and multiuser key decryption in IoT. *IEEE Internet Things Journal*, 11(1):1065–1079, 2024.

[11] Junling Guo, Lidong Han, Guang Yang, Xuejiao Liu, and Chengliang Tian. An improved secure designated server public key searchable encryption scheme with multi-ciphertext indistinguishability. *Journal of Cloud Computing*, 11:14, 2022.

[12] Mizuki Hayashi and Keita Emura. Security analysis on a public-key inverted-index keyword search scheme with designated tester. In *AsiaJCIS*, 2025, to appear.

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

[14] Zhe Jiang, Kai Zhang, Liangliang Wang, and Jianting Ning. Forward secure public-key authenticated encryption with conjunctive keyword search. *The Computer Journal,*, 66(9):2265–2278, 2023.

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

[16] 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.

[17] Hyun Sook Rhee, Jong Hwan Park, and Dong Hoon Lee. Generic construction of designated tester public-key encryption with keyword search. *Information Sciences*, 205:93–109, 2012.

[18] Nan Zhang, Baodong Qin, and Dong Zheng. Cryptanalysis of keyword confidentiality in a searchable public-key encryption scheme against malicious server. *IET Inffomation Security*, 2025(1), 2025. Article ID 2464518.