Hindawi
Security and Communication Networks
Volume 2017, Article ID 6435138, 16 pages
https://doi.org/10.1155/2017/6435138

Research Article

Multiuser Searchable Encryption with Token Freshness Verification

Dhruti Sharma[1] and Devesh C. Jinwala[2]

[1]*Sarvajanik College of Engineering and Technology, Surat, Gujarat, India*
[2]*Sardar Vallabhbhai National Institute of Technology, Surat, Gujarat, India*

Correspondence should be addressed to Dhruti Sharma; sharmadhruti77@gmail.com

Received 2 May 2017; Revised 25 September 2017; Accepted 25 October 2017; Published 26 November 2017

Academic Editor: Sherali Zeadally

A Multiuser Searchable Encryption (MUSE) can be defined with the notion of Functional Encryption (FE) where a user constructs a search token from a search key issued by an Enterprise Trusted Authority (ETA). In such scheme, a user possessing search key constructs search token at any time and consequently requests the server to search over encrypted data. Thus, an FE based MUSE scheme is not suitable for the applications where a log of search activities is maintained at the enterprise site to identify dishonest search query from any user. In addition, none of the existing searchable schemes provides security against token replay attack to avoid reuse of the same token. In this paper, therefore we propose an FE based scheme, Multiuser Searchable Encryption with Token Freshness Verification (MUSE-TFV). In MUSE-TFV, a user prepares one-time usable search token in cooperation with ETA and thus every search activity is logged at the enterprise site. Additionally, by verifying the freshness of a token, the server prevents reuse of the token. With formal security analysis, we prove the security of MUSE-TFV against chosen keyword attack and token replay attack. With theoretical and empirical analysis, we justify the effectiveness of MUSE-TFV in practical applications.

1. Introduction

With the cloud storage infrastructure, one can easily share data with multiple users at a low cost. However, maintaining security and privacy of such data located on the untrusted remote server is nontrivial [1–3]. Therefore, a common trend is to upload the encrypted data onto a third-party cloud server. However, extraction of partial information from the stored encrypted data is indeed difficult. The notion of Searchable Encryption (SE) is used to resolve the issue. In SE, a Data Owner prepares a ciphertext by associating a list of encrypted keywords (to be searched) with an encrypted payload message and uploads it onto the Storage Server. Subsequently, a Data User asks the server to search over encrypted data by issuing a search token (of keyword(s)). The server applies a token over available ciphertexts and extracts the data containing that keyword(s) (Figure 1). However, the server learns nothing else about the data while searching. Here, a payload message is encrypted using any standard encryption algorithm, whereas keywords are encrypted with the defined Searchable Encryption algorithm.

There exist numerous Searchable Encryption schemes for a single user [4–8] as well as for multiple users [9–13]. Practically, any single-user Searchable Encryption scheme can be adapted to define a multiuser Searchable Encryption scheme at the cost of a ciphertext size linear to the number of users in the system. Formally, when a single-user searchable scheme is extended to support multiple users, its ciphertext size becomes $O(U)$ for U users that subsequently raises to $O(|D| \cdot U)$ for $D = \{d1, d2, \ldots, dm\}$ data items in the system. This ultimately outputs an impractical system with $O(|D| \cdot U)$ computational overhead at the Data Owner site and $O(|D| \cdot U)$ storage overhead at the server site. As solution, several Searchable Encryption schemes in [9, 10, 14–20] with a built-in support of multiple users are devised in recent years. Amongst them, the scheme proposed by Hwang and Lee [9] is a simple extension of a single-user Searchable Encryption with the ciphertext size $O(|D| + |W| \cdot U)$, where $|W|$ is the number of keywords to be searched. However, this scheme works for the prefixed set of users. In contrast, the schemes in [10, 14–16] support the dynamic groups of users where joining/leaving a group by a member is entirely controlled

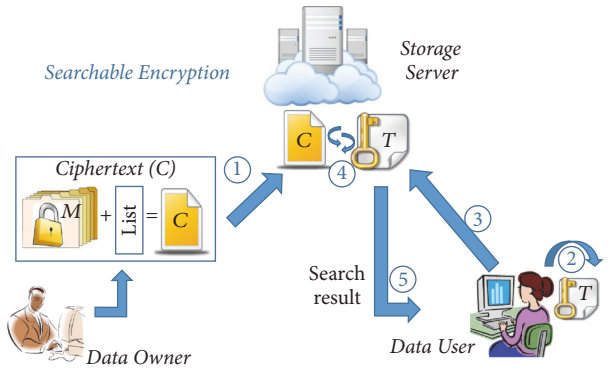

Figure 1: System model of Searchable Encryption (SE). Steps: (1) Data Owner uploads a ciphertext C (i.e., encrypted payload message + list of encrypted keywords) onto the Storage Server; (2) Data User constructs a search token T using a secret key; (3) Data User sends T to the server; (4) Storage Server applies T on C; (5) Storage Server returns the result to the requesting user.

by a Data Owner. In addition, the recent schemes in [17–20] provide Multiuser Searchable Encryption with the notion of Functional Encryption (FE) (Section 2.1) where an Enterprise Trusted Authority (ETA) is responsible for the System Setup and a master public key setup. The most notable characteristic of FE is that a system's master public key is utilized to prepare the searchable ciphertexts and a single ciphertext can serve multiple search tokens (may be issued by different users). Therefore, such FE based searchable schemes can support multiple users in the system with the optimal storage-computational overhead (i.e., $O(|D|)$) for the ciphertexts. Additionally, in the schemes [17–20], a separate search key (related to the master public key) is issued (either by an ETA or by a Data Owner) to each user. Subsequently, a user constructs a search token with an available search key. The downside is that once a user has a search key, he can prepare a search token at any time. As a result, a dishonest user colluding with the untrusted cloud server can maliciously search the valid data and the system administrator (i.e., ETA) is completely unaware about such adversarial activity. Moreover, with the existing Searchable Encryption mechanisms, there is no provision for the token freshness checking at the server site. As a result, if an unauthorized user masquerading as an authorized user has a valid token, he can use the token to make search queries in the future. In practice, there exist applications wherein every search query from the users should be logged to the enterprise trusted site in order to identify any dishonest activity performed by any user (authorized or unauthorized). In addition, there should be a provision against token replay attack to avoid misuse of a valid token. Let us take one of such applications as an example.

(i) Consider an Online Banking System, where the customers' transaction records are stored at the Bank's cloud Storage Server. Practically, these records are utilized by several official users (i.e., managers, officers, clerks, etc.) of the Bank. Let us assume that the Bank's centralized processing server (trusted authority) uses any of the existing FE based searchable schemes and accordingly issues a separate search key to each authorized user of the Bank.

In such a setup, let us take a case of a manager who is responsible for generating a daily report for the ATM transactions with a specific ATM-ID. To perform this activity, every day the manager constructs a search token (using his search key) for a query, that is, "list all ATM transactions for ATM-ID today." He issues this token to the server and collects the result. In this scenario, what happens if a peon steals the search token and masquerades as an officer to send this token to the server? In any FE based searchable scheme, the server only checks the authorization of a user. In this case, since a peon impersonates an authorized officer of the Bank, he passes the authorization test conducted by the server and gets the search result. In fact, performing such token replay attack (by reusing the token) and leaking the information about ATM transactions to the intruder (outsider) on a daily basis, the peon may provoke the criminal activities near that ATM.

From the above scenario, we say that, in the Banking system, since every search result involves critical financial information, the search activity by each user should be logged at the Bank's centralized processing server. In addition, to avoid misuse of any valid token, it is desirable to prevent token replay attack in such system.

With the existing FE based searchable schemes [17–20], a user possessing a search key can ask the server to execute a search operation at any time and therefore the search activity of a user cannot be tracked. The problem can be resolved by an interactive scheme where a search token is constructed by the centralized trusted authority on request from an authorized user. However, such solution raises the demand of secure token transmission along the entire path from the trusted authority up to the server through a user. Moreover, a token replay attack should be prevented by verifying the freshness of each search token at the server site. In addition, it is desirable to have a search operation with the support of conjunctive queries in such system.

1.1. Related Work. The notion of Searchable Encryption is introduced by Song et al. in [4] where the authors consider search over encrypted keywords within a file. However, this first practical scheme leaks the search keywords to the server and suffers from the communication overhead linear to the file size. In fact, the scheme in [4] is not secure against statistical analysis across multiple queries. To resolve the problems, Goh et al. [5] and Chang and Mitzenmacher [24] in their separate work construct the secure searchable schemes by proposing an encrypted index for a document. Though the schemes in [4, 5, 24] perform efficient search operations, they introduce storage overhead linear to the size of an index for each document. Curtmola et al. [25] propose the

first symmetric searchable encryption scheme with a formal security model. The first public key Searchable Encryption scheme is given by Boneh et al. [6] wherein a user with his private key can search over data encrypted with the corresponding public key. However, none of the schemes [4–6, 24] support conjunctive keyword search.

Conjunctive Keyword Searchable Schemes. To narrow down the scope of searching and get optimal results, several searchable schemes exist with conjunctive keyword search operation. In the symmetric key settings, Golle et al. [26] have constructed two schemes for a conjunctive keyword search. However, in the first construction of [26], the size of a capability (search token) is linear to the number of documents available on the server and so the scheme is impractical. On the other hand, the second construction of [26] is practical with a constant size capability. The other constructions based on the secret sharing and bilinear map are given by Ballard et al. [27] but they are still inefficient in terms of a size of a token linear to the number of documents being searched. In public key settings, a first conjunctive keyword searchable scheme is defined by Park et al. [8]. Subsequently, the schemes with the improved communication and storage efficiency are proposed in [9, 28]. Boneh and Waters have given a generalized scheme [29] for conjunction as well as for subset queries. Later on, a scheme with a refined form of a token (that is independent of specifying the keyword field position) is devised by Wang et al. [13]. Subsequently, B. Zhang and F. Zhang [21] have improved the security flaws of [13] and defined a conjunctive-subset keyword search. Other efficient constructions with the support of conjunctive keyword search operation are given in [22, 23, 30].

Multiuser Searchable Schemes. In public key settings, Hwang and Lee [9] have first introduced a storage efficient multiuser scheme. Subsequently, several other schemes [10, 11, 13, 14, 16] have proposed managing a group of users. However, a scheme in [11] supports the static groups of users, whereas the schemes discussed in [10, 13, 14] work for the dynamic groups of users. Apart from this, the scheme in [14] provides a single keyword search whereas the schemes in [10, 13] handle the conjunctive search queries. Recently, a multiuser multikeyword search scheme is proposed by Huang et al. [16] but its inverted index based construction cannot support an efficient conjunctive search. In addition, a scheme in [16] leaks user access control information to the server. Few other multiuser schemes [17–20] are based on the notion of FE wherein an ETA is responsible for the System Setup and a master public key setup. In these schemes, a ciphertext is prepared by a Data Owner using a master public key. A search token is constructed by a user with his own search key issued either by the ETA as in [18–20] or by the Data Owner as in [17]. A scheme in [17] offers a constant size ciphertext and a constant size token. However, the scheme [17] is computationally inefficient since, to encrypt an index for a document, the encryption algorithm involves a computational complexity linear to the number of authorized users for that document. In a scheme of [18], the Storage Server has a list of authorized users (U_List), and thus each enrollment/revocation of a user is known to the server. This indeed leaks information about users (i.e., a number of users in the system, the users' activity) to the Storage Server. The other two schemes [19, 20] use CPABE (Ciphertext Policy Attribute Based Encryption) to manage access control of users. However, amongst all these schemes, only the schemes in [9, 10, 13, 16] support multikeyword (specifically conjunctive) search and multiple users at the same time. There is no FE based scheme proposing a conjunctive keyword based search.

Secure Channel-Free Searchable Schemes. There exist searchable schemes in [7, 31, 32] with secure channel-free architecture for a token transmission. However, these schemes support a single keyword search. The most recent conjunctive search schemes [30, 33] provide a secure channel-free token transmission.

To the best of our knowledge, none of the existing schemes define a secure channel-free conjunctive keyword based Searchable Encryption that prevents token replay attack in multiuser environment.

1.2. Our Contributions. In this paper, we propose a Multiuser Searchable Encryption with Token Freshness Verification (MUSE-TFV). In MUSE-TFV, a user constructs a search token in cooperation with the ETA and thus every search activity from each user is logged at the enterprise trusted site. Moreover, each search token is one-time usable token. The server avoids reuse of the same token by verifying the freshness of the token using a verification key given by the ETA. Our main contributions are as follows.

(i) *Multiuser Support*. Utilizing the notion of FE, we devise a Searchable Encryption scheme that supports multiple users, with a constant size ciphertext (i.e., independent of the number of users). Our scheme has an optimal computational overhead at the Data Owner site and an optimal storage overhead at the server site.

(ii) *Token Freshness Verification*. We propose a token freshness verification at the server site by adapting Haller's S/Key One-Time Password System [34] and prevent token replay attack from the system.

(iii) *Conjunctive Keyword Search*. With the proposed scheme, we offer a conjunctive keyword search with a constant sized search token.

(iv) *Secure Channel-Free Architecture*. We offer a secure channel-free architecture to transfer a token securely via any public channel without channel setup overhead.

(v) *Theoretical Analysis and Empirical Evaluation*. We present a detailed theoretical analysis to show the efficiency of the proposed scheme. Additionally, with experimental evaluation of MUSE-TFV for different size system (with a different number of keywords) and different number of users, we justify its effectiveness.

1.3. Organization of the Rest of the Paper. The rest of the paper is organized as follows: In Section 2, we briefly discuss the

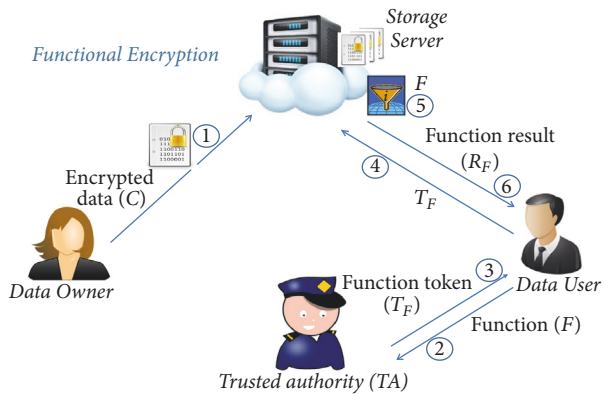

FIGURE 2: System model of Functional Encryption (FE). Steps: (1) Data Owner uploads ciphertext C onto the Storage Server; (2) Data User requests TA for a token of a function (F); (3) TA issues a token T_F to the user; (4) Data User sends T_F to the Storage Server; (5) Storage Server runs F on available C; (6) Storage Server forwards the result R_F to the user.

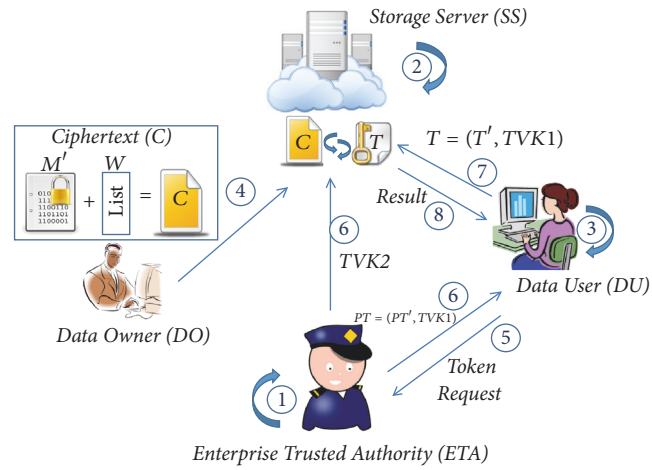

FIGURE 3: System model of MUSE-TFV.

preliminaries required for the proposed scheme. In Section 3, we define the formal model of MUSE-TFV, the proposed algorithms, and the attack model with security definition. We elaborated the algorithms with a detailed security analysis in Section 4. Further, in Section 5, we present a theoretical analysis and empirical evaluation of MUSE-TFV. Finally, we put the concluding remarks in Section 6.

2. Preliminaries

In this section, we present an overview of a Functional Encryption, a cryptographic primitive (i.e., Bilinear Map), and a hardness assumption associated with the proposed scheme.

2.1. Functional Encryption (FE). FE is a generalization of the existing access control mechanisms, namely, Identity Based Encryption (IBE) [35, 36], Attribute Based Encryption (ABE) [37–39], and Predicate Encryption (PE) [29, 40]. In FE, apart from the Data Owner, Data User, and the Storage Server, there exists an additional centralized trusted authority (TA) that is responsible for the System Setup and generation of a master public-private key pair. A Data Owner prepares the ciphertexts with a master public key and stores them to the Storage Server. To execute a predefined function at the server site, a user asks the TA for the corresponding token. In response, the TA constructs a token utilizing a master private key and issues it to the user. The server runs the function on the availability of a token from a user and sends the result to the user (Figure 2). In such a setup, any user who possesses a token can ask the server for the function execution. Since the server could use the same set of ciphertexts to execute a function with different tokens (may be from different users), we say that the FE supports multiple users in the system.

2.2. Bilinear Map. Bilinear map is a mathematical tool for pairing based cryptography. It is defined using suitable cryptographic groups. Let G_1 and G_2 be two multiplicative cyclic groups of prime order p. For these groups, a bilinear map $e: G_1 \times G_1 \rightarrow G_2$ must satisfy the following properties:

(1) *Bilinear*: given random $P, Q \in G_1$ and $a, b \in Z_p^*$ we have $e(aP, bQ) = e(P, Q)^{ab}$.

(2) *Nondegenerate*: if P is a generator of G_1, then $e(P, P)$ is a generator of G_2.

(3) *Computable*: given $P, Q \in G_1$, there exists a polynomial time algorithm to compute $e(P, Q) \in G_2$.

2.3. Hardness Assumption

Decisional Diffie-Hellman (DDH) Assumption. Let G_1 be a cyclic group of prime order p and P is a generator of G_1. The *Decisional Diffie-Hellman* problem is to distinguish the tuple (aP, bP, abP) from (aP, bP, cP) for any random $a, b, c \in Z_p^*$. Let us assume that the DDH problem is (ϵ, t)-hard in G_1. Then there does not exist any polynomial time (t) adversary $\mathcal{A}$ that can solve the DDH problem with a nonnegligible advantage ϵ, if $|Pr[\mathcal{A}(aP, bP, abP)] - Pr[\mathcal{A}(aP, bP, cP)]| \leq \epsilon$.

3. Proposed Multiuser Searchable Encryption with Token Freshness Verification (MUSE-TFV)

We list out the notations used throughout the paper in Notations section. We include a system model, the associated algorithms, and the attack model with the security definition for the proposed scheme.

3.1. System Model. The proposed MUSE-TFV involves four entities: (i) Data Owner (DO), (ii) Data User (DU), (iii) Storage Server (SS), and (iv) Enterprise Trusted Authority (ETA) (Figure 3).

The interactive actions amongst these entities are as follows:

(1) Initially, the ETA sets up the system's public parameters and a master secret key.

(2) Using public parameters, the SS computes a public-private key pair (Y, y) and publishes Y while keeping y secret.

(3) Using public parameters, the DU computes a public-private key pair (X, x) and publishes X while keeping x secret.

(4) A DO prepares a ciphertext (C) by associating an encrypted payload (M') with a list W of encrypted keywords and uploads it onto the SS. All the keywords in the list are encrypted with an *Encryption()* algorithm of proposed MUSE-TFV.

(5) To execute a search operation, the DO requests the ETA for a token of a conjunctive query.

(6) The ETA computes a token (PT') and corresponding token verification keys $(TVK1, TVK2)$. The ETA issues a partial token $PT = (PT', TVK1)$ to the DU and $TVK2$ to the SS.

(7) The DU constructs a search token (T') from PT' and issues a final token $T = (T', TVK1)$ to the SS over a public channel.

(8) The proposed *Search()* algorithm is executed on the server SS. With the available $(TVK1, TVK2)$, the SS checks the token freshness. The SS applies the fresh token T' on the available C. If C satisfies the token T, the algorithm outputs a result $R = (E_y(E_X(M')))$; otherwise it outputs $\perp$. The algorithm applies T on all available C and generates the corresponding R.

Note. Steps (2), (3), and (4) can run in parallel.

Assumptions. (i) The payload $M' = E_{key}(M)$, where E is any symmetric encryption cipher with a symmetric key *key*. (ii) All DUs are authorized by the ETA. At the time of authorization, ETA issues (pp, key) to the DU. (iii) Before issuing a partial token PT, the ETA checks the authenticity of a DU with any standard authentication protocol. (iv) The SS is a semihonest server; that is, it follows the system protocol but tries to breach data privacy. (v) There exists a secure channel between the ETA and the SS. (vi) The $TVK2$ is stored in a system table of the SS. The size of the system table is linear to the number of DUs.

3.2. Algorithms. The proposed MUSE-TFV involves the following polynomial time algorithms:

(1) **Setup**(α, n). The *Setup* algorithm runs by the ETA. The algorithm takes a security parameters α and n as inputs. The algorithm outputs the system's public parameter pp and a master secret key msk. It defines a keyword space $\mathcal{KS}$ for n keywords.

(2) **SKeyGen**(pp). The *Server Key Generation* algorithm runs by the server SS. The algorithm takes the system's public parameter pp as inputs. It selects a random $y \in Z_p^*$ and computes the public-private key pair (Y, y) for the server SS.

(3) **UKeyGen**(pp). The *User Key Generation* algorithm runs by the DU. The algorithm takes the system's public parameter pp as inputs. It selects a random $x \in Z_p^*$ and computes the public-private key pair (X, x) for the Data User, DU.

(4) **Encryption**(pp, W, Y, M'). The *Encryption* algorithm runs by the DO. The algorithm constructs a ciphertext C' from the list of keywords $W = \{w_1, w_2, \ldots, w_n\}$ using pp and Y. It associates C' with an encrypted payload M' and outputs a ciphertext $C = (C', M')$.

(5) **TokGen**(pp, msk, Q, X, x, Y). The *Token Generation* is an interactive algorithm where initially a DU supplies a conjunctive query $Q = (W', I')$ to the ETA. Here, $W' = \{w'_1, w'_2, \ldots, w'_t\}$ is a set of keywords and $I' = \{I_1, I_2, \ldots, I_t\}$ shows their positions in $\mathcal{KS}$. For each new query, the ETA assigns a unique token identification string $(TOKID)$ in order to generate the token verification keys $(TVK1, TVK2)$. Subsequently, the ETA constructs a token PT' using msk and X. The ETA then issues a partial token $PT = (PT', TVK1)$ to the DU and $(TVK2)$ to the SS. With an available PT, the DU constructs T' and outputs a final token $T = (T', TVK1)$.

(6) **Search**(C, T, y). The *Search* algorithm runs by the SS. The algorithm utilizes $(TVK1, TVK2)$ to verify the freshness of T. If T is fresh, the algorithm performs a conjunctive search using (T', C', y). It returns the result $R = (E_y(E_X(M')))$ to the DU if C' satisfies the conjunctive query Q within T'; otherwise it returns $\perp$. The algorithm applies T' on all the ciphertexts. At last, the algorithm updates the system table entry of $TVK2$ for the requesting DU to prevent a token replay attack.

The algorithms involved in the verification key generation and token verification as well as system table update are discussed in Section 4.2.

3.3. Flowchart. To show the process of the proposed MUSE-TFV, we define four phases: (i) System Setup, (ii) Data Upload, (iii) Token Generation, and (iv) Search. The sequence of the proposed algorithms utilized by the entities (i.e., ETA, DO, DU, SS) during each of these phases is given as a flowchart in Figure 4. As shown in Figure 4(a), all four entities are involved in System Setup phase where a public parameter (pp) and various keys (i.e., $msk, key, (X, x), (Y, y)$) are defined. On the other hand, Data Upload phase (Figure 4(b)) includes only DO and SS since, during this phase, a DO prepares a ciphertext C and uploads it on to the SS. The interactive steps amongst DU, ETA, and SS during Token Generation phase are shown in Figure 4(c) wherein initially a DU sends a conjunctive query Q to the ETA. In response, the ETA sends a partial token along with a token verification key (i.e., $(PT, TVK1)$) to the DU. In addition, the ETA sends a token verification key (i.e., TVK2) to the SS. With the available $(PT, TVK1)$, the DU prepares a final token T. During Search phase, the DU sends T to the SS as shown in Figure 4(d). In response, the SS finds the results R for the available ciphertexts and forwards these results to the DU.

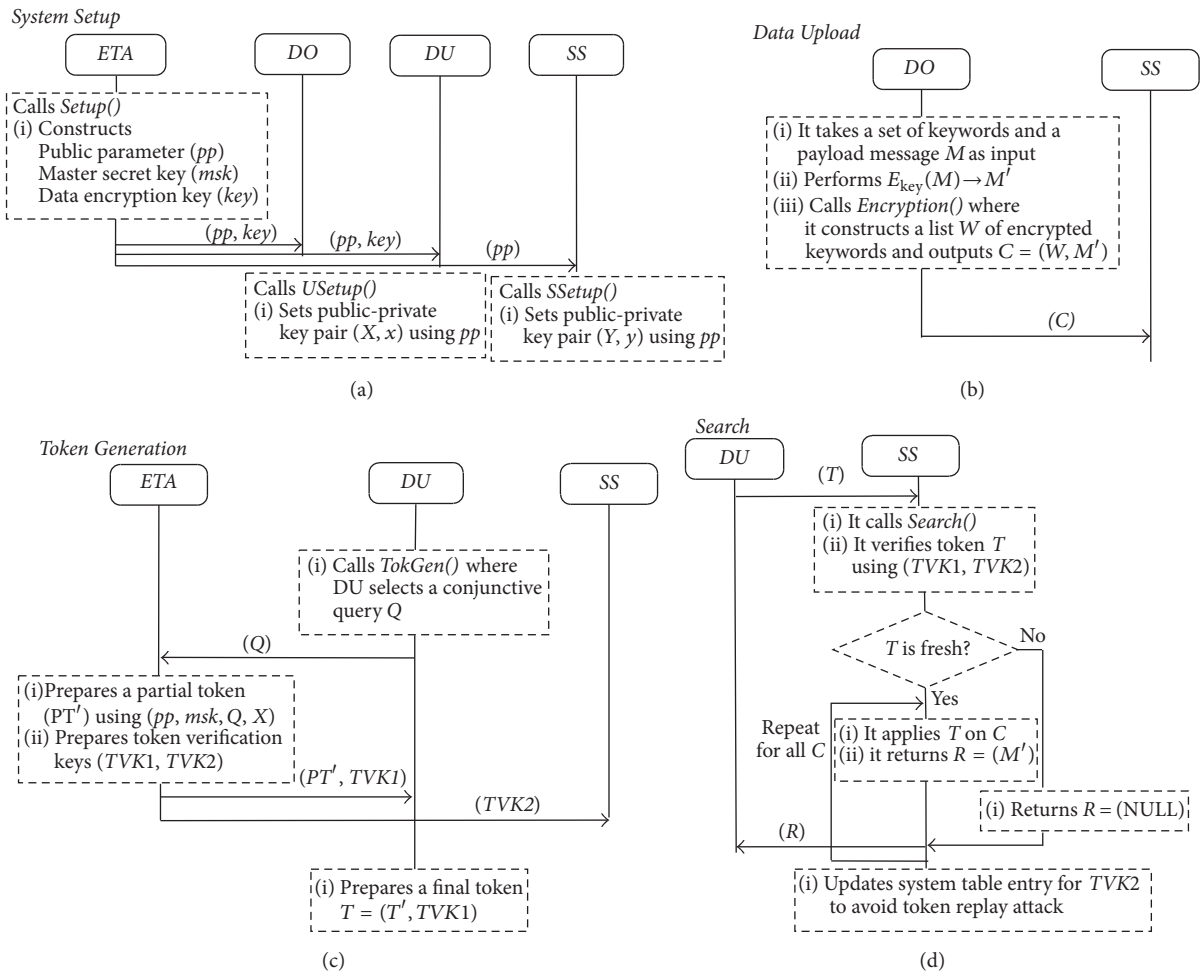

Figure 4: Flowchart of MUSE-TFV.

3.4. Attack Model and Security Definitions. First, we reemphasize that the principal motivation of the proposed MUSE-TFV is to overcome the limitation in the existing Searchable Encryption schemes that allow replay of tokens and thus lack verification of token freshness. Thus, MUSE-TFV is aimed at supporting a Searchable Encryption scheme with the novel provision for verification of the token freshness and thereby avoiding replay attacks. Therefore in the attack model described here we consider only token replay attacks and assume that any other attack against the scheme can be mitigated by using already existing mitigation approaches.

We assume that an adversary $\mathscr{A}$ has the capabilities to perform the following attacks:

(1) The server SS as an adversary $\mathscr{A}$ can perform chosen keyword attack to deduce the plaintext (keywords) from the available ciphertexts (lists of encrypted keywords) and tokens.

(2) The Data User, DU, as an adversary $\mathscr{A}$ can perform token replay attack to reuse the maliciously captured token.

With SS as an adversary, we define semantic security (a.k.a. indistinguishability against chosen keyword attack (IND-CKA)) for the proposed conjunctive keyword search scheme based on the security game ICLR (Indistinguishability of Ciphertext from Limited Random) [26, 41] as follows.

Definition 1 (ICLR). Let $\mathscr{A}$ be a polynomial bounded adversary and $\mathscr{B}$ be a challenger. With ICLR, when $\mathscr{A}$ has issued a keyword set W and a subset $T \subseteq \{1, 2, \ldots, n\}$, $\mathscr{B}$ responds with two encrypted keyword sets associated with T in such a way that $\mathscr{A}$ cannot distinguish the encrypted keyword sets created with T. Thus, with this game, we achieve our security goal where we require that $\mathscr{A}$ should not be able to deduce the plaintext from other keyword sets. The following are the steps for the game ICLR [26, 41].

(1) $\mathscr{A}$ adaptively requests $\mathscr{B}$ for the Encryption (pp, W_i, Y, M') of any keyword set W_i and any search token.

(2) $\mathscr{A}$ selects a keyword set W, a subset $T \subseteq \{1, 2, \ldots, n\}$, and $t \in T$ in such a way that none of the tokens given in Step (1) are distinguishing for $Rand(W, T)$ and $Rand(W, T - \{t\})$. Here, $Rand(W, T)$ outputs a set W where the keywords indexed by T (i.e., the set $\{w_i \mid i \in T\}$) are replaced by random values. $\mathscr{A}$ then sends (W, T, t) to the challenger $\mathscr{B}$.

(3) $\mathcal{B}$ constructs two keyword sets $W_0 = Rand(W, T-\{t\})$ and $W_1 = Rand(W, T)$. $\mathcal{B}$ then randomly chooses $b \in \{0, 1\}$ and returns Encryption (pp, W_b, Y, M') to $\mathcal{A}$.

(4) $\mathcal{A}$ again makes requests for encrypted keyword sets and search tokens, with the restriction that he cannot ask for the token that is distinguishing for W_0 and W_1.

(5) $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$ and wins the ICLR game if $b' = b$.

We say that the polynomial time adversary $\mathcal{A}$ has an advantage ϵ in this attack game, if

$$Adv_{\mathcal{A}}(1^{\alpha}) = \left| Pr\left[b' = b\right] - \frac{1}{2} \right| > \epsilon. \quad (1)$$

Additionally, we define the security against token replay attack based on the following actions performed by a Data User, DU, as an adversary $\mathcal{A}$.

(1) $\mathcal{A}$ intercepts a token $T = (T', c = E_Y(TVK1))$ transmitted from the ETA to the DU (or from a DU to the SS) and stores it.

(2) To reuse the token T, $\mathcal{A}$ replaces its verification key part, that is, $c = E_Y(TVK1)$, with c' in such a way that the SS considers a forged $T = (T', c')$ as a fresh token and returns a result R.

(3) $\mathcal{A}$ repeats Step (2) till he does not receive the result R.

We say that an adversary $\mathcal{A}$ is successful in token replay attack if he gets the result R using a forged value of c.

4. Construction of MUSE-TFV

In this section, we give the formal construction for the proposed algorithms of MUSE-TFV. We also present a token verification procedure used in the design of the MUSE-TFV. Additionally, we provide a security analysis for the proposed scheme.

4.1. Formal Construction. The concrete constructions for the proposed algorithms are as follows.

(1) **Setup**(α, n). Let G_1 and G_2 be bilinear groups of prime order p where a security parameter α defines the group size. Let $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear pairing and $H_1 : \{0, 1\}^* \rightarrow {Z_p}^*$ is a hash function. Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^b$ be any standard hash function (e.g., SHA2) that outputs a message digest of b bits. Let P be a generator of G_1. The algorithm initializes the keyword space $\mathcal{KS}$ of total n keywords. For each jth keyword, it randomly selects $k_j \in {Z_p}^*$ and computes $K_j = k_j P$. Finally, the algorithm sets the public parameter $pp = \{H_1, H, G_1, G_2, P, e, \{K_j\}_{1 \le j \le n}\}$ and a master secret key $msk = \{\{k_j\}_{1 \le j \le n}\}$.

(2) **SKeyGen**(pp). The algorithm selects a random $y \in {Z_p}^*$ and computes $Y = yP$. It sets the public-private key pair for the server SS as (Y, y).

(3) **UKeyGen**(pp). The algorithm selects a random $x \in {Z_p}^*$ and computes $X = xP$. It sets the public-private key pair for the user DU as (X, x).

(4) **Encryption**(pp, W, Y, M'). The algorithm takes as input a list of keywords $W = \{w_1, w_2, \ldots, w_n\}$. It chooses a random $r_1 \in {Z_p}^*$ and constructs a ciphertext $C' = \{C_{1j}, C_2\}_{1 \le j \le n}$, where $C_{1j} = r_1(H_1(w_j)P + K_j) + r_1 Y$, $C_2 = r_1 P$. Finally, it outputs a ciphertext $C = (C', M')$, where M' is an encrypted payload.

(5) **TokGen**(pp, msk, Q, X, x, Y). This interactive algorithm works in 3 phases.

(a) A DU sends a conjunctive query $Q = (W', I')$ to the ETA where $W' = \{w'_1, w'_2, \ldots, w'_t\}$ is a set of keywords and $I' = \{I_1, I_2, \ldots, I_t\}$ is a set of positions of keywords in $\mathcal{KS}$.

(b) In response, the ETA chooses a unique token identification string $TokID \in \{0, 1\}^{\ell}$ and a secret random integer N. The ETA uses $TokVerKey(TokID, N, H()) \rightarrow (TVK1, TVK2)$ algorithm to construct the token verification keys. The ETA selects $t_1 \in Z_p^*$ randomly. It uses msk and X to construct a token component $PT' = \{PT_1, PT_2\}$, where $PT_1 = t_1(\sum_{j=I1}^{It}(H_1(w_j) + k_j))P + t_1 X$, $PT_2 = t_1 P$. At last, the ETA sends a partial token $PT = (PT', E_Y(TVK1))$ to the DU. At the same time, it forwards $(E_Y(TVK2))$ to the SS.

(c) The DU selects a random element $a' \in Z_p^*$. Using x and Y, the DU computes $T' = \{T_1, T_2, T_3, T_4\}$ as follows:

$T_1 = \tau + a'Y$, $T_2 = PT_2 = t_1 P$, $T_3 = a'P$, $T_4 = I'$

Where $\tau = PT_1 - xPT_2 = t_1(\sum_{j=I1}^{It}(H_1(w_j) + k_j))P$.

Finally, the algorithm outputs a token $T = (T', E_Y(TVK1))$.

(6) **Search**(C, T, y). The algorithm applies $D_y(E_Y(TVK1))$ and $D_y(E_Y(TVK2))$ to get the original verification key $(TVK1, TVK2)$ from the encrypted values using a private key y of the SS. The algorithm then calls $TokVer(TVK1, TVK2)$ to verify the freshness of the input token T. If a token is fresh (i.e., $TokVer(\cdot) \rightarrow 1$), it applies T' of T on an available ciphertext C' from C as follows.

The algorithm computes

$$\tau_1 = \sum_{j=I1}^{It} \left(C_{1j} - yC2\right)$$

$$= \sum_{j=I1}^{It} \left(r_1\left(H_1\left(w_j\right)P + K_j\right) + r_1 Y - yr_1 P\right)$$

$$= r_1\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)P + K_j\right)\right),$$

$$\begin{aligned}\tau_2 &= T_1 - yT_3 \\ &= t_1\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)+k_j\right)\right)P + a'Y - ya'P \\ &= t_1\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)+k_j\right)\right)P.\end{aligned} \tag{2}$$

Then, it checks the following correctness:

$$e\left(\tau_1, T_2\right) = e\left(\tau_2, C_2\right). \tag{3}$$

If (3) is satisfied, then the algorithm outputs the associated payload message M'; as a result $R = E_y(E_X(M'))$. Here, encryption with a public key X of DU provides confidentiality and signature with the private key y of SS maintains integrity of a result R during transit. The algorithm repeatedly applies T' on each available ciphertext at the server SS. At last, the algorithm updates the current entry of $TVK2$ in the system table with $TUpdate(TVK2, TVK1)$.

Note. (i) The algorithms *TokVerKey*(), *TokVer*(), and *TUpdate*() are described in Section 4.2. (ii) The query Q from a DU to the ETA is in plaintext format. It does not impact the security of token as even if any unauthorized DU maliciously captures a partial token, he is unable to construct a final token unless having secret key x. (iii) The E/D for the verification keys is any standard encryption/decryption cipher. The encryption of the verification keys with the public key Y of SS prevents their modification by a malicious DU.

Correctness. LHS of (3):

$$\begin{aligned}e\left(\tau_1, T_2\right) &= e\left(r_1\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)P + K_j\right)\right), t_1P\right) \\ &= e\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)P + K_j\right), P\right)^{r_1t_1} \\ &= e\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)P + k_jP\right), P\right)^{r_1t_1} \\ &= e\left(P, P\right)^{r_1t_1\Delta}.\end{aligned} \tag{4}$$

RHS of (3):

$$\begin{aligned}e\left(\tau_2, C_2\right) &= e\left(t_1\left(\sum_{j=I1}^{It}\left(H_1\left(w_j\right)+k_j\right)\right)P, r_1P\right) \\ &= e\left(P, P\right)^{r_1t_1\Delta}.\end{aligned} \tag{5}$$

Here, $\Delta = \sum_{j=I1}^{It}(H_1(w_j) + k_j)$. From (4) and (5), the correctness is proved.

4.2. Token Verification Procedure. To define a token verification procedure, we borrow the idea from Haller's S/Key One-Time Password System [34]. The S/Key scheme provides a technique to construct a one-time password at the client site and its verification at the host site. The scheme works on 3 parameters $(s, RN, H())$, where s is a secret string, RN represents the number of times the hash is applied on s, and $H()$ is any standard cryptographic hash function. We adopt similar parameters to define a token verification procedure for the proposed MUSE-TFV. The token freshness verification involves three algorithms:

(1) **TokVerKey**$(\mathbf{s}, \mathbf{RN}, \mathbf{H}())$: the *token verification key generation* algorithm outputs two keys (K_1, K_2), where $K_1 = H^{RN}(s)$ and $K_2 = H^{RN-1}(s)$.

(2) **TokVer**$(\mathbf{K_1}, \mathbf{K_2})$: the *token verification* algorithm verifies the freshness of a token by checking $K_1 = H(K_2)$. If condition is true, the algorithm outputs "1" otherwise "0."

(3) **TUpdate**$(\mathbf{K_1}, \mathbf{K_2})$: the *token update* algorithm updates the current memory location of K_1 with K_2; that is, it performs $K_1 = K_2$.

The original S/Key mechanism is defined with the traditional hash function, that is, MD4. For MUSE-TFV, we prefer SHA-2 to avoid collision attack.

4.3. Security Analysis. We analyze the semantic security of MUSE-TFV against chosen keyword attack (IND-CKA) under DDH assumption. Additionally, we prove that the proposed MUSE-TFV provides security against token replay attack.

Theorem 2. *The proposed MUSE-TFV is semantically secure against a server SS as an adversary according to the game ICLR, assuming DDH is intractable.*

Proof. Let us assume a server SS as an adversary $\mathcal{A}$ can attack the proposed scheme in a polynomial time. Suppose $\mathcal{A}$ makes at most q_k token queries where $q_k < p$ and has the advantage ϵ in solving DDH problem in G_1. Let G_1 and G_2 be two groups of prime order p and P be the generator of G_1. We build a simulator $\mathcal{B}$ as a challenger that has the advantage $\epsilon' = \epsilon/e^n q_k n$ to simulate the game where e is base of natural logarithm.

Suppose an instance (aP, bP, cP) of the DDH problem in G_1 is the $\mathcal{B}$'s challenge information where $a, b, c \in Z_p^*$. The goal of $\mathcal{B}$ is to distinguish $cP = abP$ from random element in G_1. One restriction is that the random element z is independent of the location t selected in ICLR game; then the simulation game is demonstrated as follows.

(1) **Setup**. An adversary $\mathcal{A}$ randomly selects $y \in Z_p^*$ and computes $Y = yP$. $\mathcal{A}$ then defines a public-private key pair (Y, y). Let $(X = xP, x)$ be the $\mathcal{B}$'s public-private key pair.

(2) **Encryption Queries**. An adversary $\mathcal{A}$ issues the queries for the ciphertext of the keyword set $W_i = \{w_{i1}, w_{i2}, \ldots, w_{in}\}$. In response, challenger $\mathcal{B}$ simulates $Encryption(pp, W_i, Y)$ as follows.

(i) $\mathscr{B}$ selects $\gamma_j \in Z_p^*$ for each keyword $w_{ij} \in W_i$, where $1 \le j \le n$.

(ii) $\mathscr{B}$ chooses a random value $r_{1i} \in Z_p^*$ and constructs a ciphertext $C_i = \{(C_{1j})_i, (C_2)_i\}_{1 \le j \le n}$, where

$$\begin{aligned}
(C_{11})_i &= r_{1i}(\gamma_1 P + K_1) + r_{1i} Y \\
(C_{12})_i &= r_{1i}(\gamma_2 P + K_2) + r_{1i} Y \\
&\vdots \\
(C_{1z})_i &= br_{1i}(\gamma_z P + K_z) + r_{1i} Y \\
&\vdots \\
(C_{1n})_i &= r_{1i}(\gamma_n P + K_n) + r_{1i} Y \\
(C_2)_i &= r_{1i} P.
\end{aligned} \tag{6}$$

(3) **Token Queries**. To evaluate *Search()* algorithm, $\mathscr{A}$ issues the token queries by sending $Q_i = (W_i', I_i')$, where $W_i' = \{w_{iI1}', w_{iI2}', \ldots, w_{iIt}'\}$ and $I_i' = \{I1, I2, \ldots, It\}$ to $\mathscr{B}$. $\mathscr{B}$ takes a partial token $PT' = \{PT_1, PT_2\}$ from the ETA where $PT_1 = t_1(\sum_{j=I1}^{It}(H_1(w_j) + k_j))P + t_1 X$, $PT_2 = t_1 P$. $\mathscr{B}$ then selects a random $a' \in Z_p^*$ and computes final token $T_i = \{T_{1i}, T_{2i}, T_{3i}, T_{4i}\}$ as follows.

$T_{1i} = \tau + a'Y$, $T_{2i} = t_1 P$, $T_{3i} = a'P$, $T_{4i} = I'$, where $\tau = PT_1 - xPT_2 = t_1(\sum_{j=I1}^{It}(H_1(w_j) + k_j))P$.

At last, $\mathscr{B}$ sends this token T_i to $\mathscr{A}$.

(4) **Challenge**. $\mathscr{A}$ issues a tuple (W_i, T, t) to $\mathscr{B}$ where $T \subseteq \{1, n\}$ and $t \in T$.

If $z \ne t$, $\mathscr{B}$ sends a random guess as the response to the DDH challenge.

If $z = t$, $\mathscr{B}$ responses are as follows.

(a) It first sets $h_t = c(\gamma_t P + K_t) + cY$.
(b) It sets $h_{1j} = \ell_j$, for $j \ne t$, $j \in T$, where $\ell_j \in Z_p^*$.
(c) It sets $h_{1j} = a(\gamma_j P + K_j) + aY$, for $j \ne t$, $j \notin T$.
(d) It sets $h_2 = aP$.

Finally, $\mathscr{B}$ sends $\{h_{1j}, h_2\}$ for $1 \le j \le n$ as challenge ciphertext to $\mathscr{A}$.

If $z = t$, then $\mathscr{B}$ wins the security game. The ciphertext for every position $j \notin T$ is the encryption of W and ciphertext in position t where $c = ab$ is also an encryption of W. Otherwise, for other position, it is not.

(5) **More Queries**. $\mathscr{A}$ queries encryption of other keyword sets and tokens that $\mathscr{A}$ has not asked before. $\mathscr{B}$ responds in the same way as in Step (2) and Step (3). The restriction is that $\mathscr{A}$ cannot issue the aforementioned queries for location t.

(6) **Guess**. At the end, $\mathscr{A}$ outputs the guess $b' \in \{0, 1\}$. If $b' = 1$ and B outputs "Yes," then (aP, bP, cP) is considered as a DDH tuple. Thus, for $z = t$, we can prove that (aP, bP, cP) is a DDH tuple as follows.

We know from (3) that

$$e(\tau_1, T_2) = e(\tau_2, C_2). \tag{7}$$

This can be represented as

$$\begin{aligned}
e(\tau_1, T_2) &= e(br_1(\tau_z P + K_z), t_1 P) \\
&= e(P, P)^{br_1(\tau_z + K_z)t_1}, \\
e(\tau_2, C_2) &= e(t_1 H_1(w_z + k_z) P, r_1 P) \\
&= e(P, P)^{r_1 H_1(w_z + k_z)t_1}.
\end{aligned} \tag{8}$$

From (8), we get

$$e(P, P)^{br_1(\tau_z + K_z)t_1} = e(P, P)^{r_1 H_1(w_z + k_z)t_1}. \tag{9}$$

Now, from the challenge ciphertext,

$$\begin{aligned}
e(\tau_1, T_2) &= e(c(\tau_t P + K_t), t_1 P) = e(P, P)^{c(\tau_t + K_t)t_1}, \\
e(\tau_2, C_2) &= e(t_1 H_1(w_t + k_t) P, aP) \\
&= e(P, P)^{aH_1(w_t + k_t)t_1}.
\end{aligned} \tag{10}$$

From (10), we get

$$e(P, P)^{c(\tau_t + K_t)t_1} = e(P, P)^{aH_1(w_t + k_t)t_1}. \tag{11}$$

Now, from (9) and (11)

$$\begin{aligned}
&\frac{e(P, P)^{br_1(\tau_z + K_z)t_1}}{e(P, P)^{r_1 H_1(w_z + k_z)t_1}} = \frac{e(P, P)^{c(\tau_t + K_t)t_1}}{e(P, P)^{aH_1(w_t + k_t)t_1}} \\
&\therefore e(P, P)^{br_1(\tau_z + K_z)t_1} e(P, P)^{aH_1(w_t + k_t)t_1} \\
&\quad = e(P, P)^{c(\tau_t + K_t)t_1} e(P, P)^{r_1 H_1(w_z + k_z)t_1} \\
&\therefore e(P, P)^{abr_1(\tau_z + K_z)t_1} e(P, P)^{H_1(w_z + k_z)t_1} \\
&\quad = e(P, P)^{cr_1(\tau_z + K_z)t_1} e(P, P)^{H_1(w_z + k_z)t_1} \\
&e(P, P)^{abr_1(\tau_z + K_z)t_1} = e(P, P)^{cr_1(\tau_z + K_z)t_1} \\
&e(P, abP) = e(P, cP) \\
&ab = c.
\end{aligned} \tag{12}$$

On the other hand, if $b' = 0$, we cannot prove that the challenge (cP, bP, cP) is a DDH tuple, since encryption at position j is random and it cannot confirm (12). However, the advantage of $\mathscr{A}$ to win the game ICLR is same as that of the $\mathscr{B}$ which solves the DDH challenge.

Now, the following are the two simulations of $\mathscr{B}$'s advantages.

(i) $S1$: $\mathscr{B}$ responds to the search token queries for n keyword issued by $\mathscr{A}$.

(ii) $S2$: $\mathscr{B}$ is not aborted in the challenge phase.

For large enough q_k, the probability of $S1$ and $S2$ can be defined as

$$Pr\,[S1] = \frac{1}{e^n}, \qquad Pr\,[S2] = \frac{1}{q_k n}. \tag{13}$$

Thus, the $\mathscr{B}$'s advantage ϵ' in solving the DDH problem is $\epsilon' = \epsilon \cdot Pr[S1 \cap S2] \geq \epsilon/e^n q_k n$.

According to Propositions 1 and 2 of [26], if there exists an adversary with nonnegligible advantage to win ICC game, then there exists another adversary with a nonnegligible advantage to win the ICLR game. However, as per the above proof, the advantage of $\mathscr{B}$ is $\epsilon/e^n q_k n \in [0, 1/2e^n q_k n]$ which is negligible. Thus, the proposed MUSE-TFV scheme is at least $(1 - 1/2e^n q_k n)$ secure under the ICLR game if DDH assumption is intractable. This completes the proof for Theorem 2. □

Theorem 3. *The proposed MUSE-TFV provides security against token replay attack.*

Proof. Let us assume a DU as an adversary $\mathscr{A}$ can perform a token replay attack as follows.

(1) An adversary $\mathscr{A}$ maliciously captures a valid token $T = (T', c = E_Y(TVK1))$ and stores it.

(2) To reuse the token T, an adversary $\mathscr{A}$ replaces its verification key part, that is, $c = E_Y(TVK1)$, with c' in such a way that the further execution of *TokVer()* (at the site of SS) outputs "1" and so the SS returns a result R.

If $Csize$ is the size of a ciphertext generated by an encryption algorithm E, then an adversary $\mathscr{A}$ required 2^{Csize} attempts to forge a value "c." With any standard secure algorithm (i.e., 160-bit ECEL (ECC based Elgamal Encryption) (as public key Y of SS is an element from a group of points of an elliptic curve, any ECC based encryption algorithm must be used)), the probability of an adversary $\mathscr{A}$ to guess a valid $(c' = c)$ is $1/(2^{160})$.

Additionally, the adversary $\mathscr{A}$ is completely unaware about the other verification key $TVK2$ available at the site of the SS. Thus, a token with the replaced verification key, that is, $T = (T', c')$, must be issued to the SS to check the output of $TokVer(TVK1, TVK2)$ algorithm. Denoting CC as a communication cost (from a DU to SS) of a single message, we find $O(CC \cdot 2^{160})$ communication complexity in the system for 2^{160} attempts potentially performed by an adversary $\mathscr{A}$ to forge a value of c.

However, with a communication link of 100 Mbps and a Maximum Transmission Unit (MTU) of 1500 bytes (Ethernet), it requires about $57 \cdot 10^{30}$ years to attempt all the possible values of c. Thus, for any adversary $\mathscr{A}$, the probability of getting the result R by forging the value c is negligible.

Thus, we say that the proposed scheme MUSE-TFV is secure against token replay attack. □

Table 1: Comparative analysis: significant characteristics.

Schemes	MU	MKC	MKQ	SCF	TFV
Hwang and Lee 2007 [9]	√	√	√	×	×
Kiayias et al. 2016 [17]	√	×	×	×	×
Zhang et al. 2016 [18]	√	√	×	×	×
Wang et al. 2016 [19]	√	×	×	×	×
B. Zhang and F. Zhang 2011 [21]	×	√	√	×	×
Ding et al. 2012 [22]	×	√	√	√	×
Chen et al. 2012 [23]	×	√	√	×	×
MUSE-TFV	√	√	√	√	√

MU: multiuser support, MKC: multiple keywords in ciphertext, MKQ: multiple keywords in query, SCF: secure channel-free architecture, and TFV: token freshness verification.

5. Theoretical Analysis and Empirical Evaluation

In this section, we first present theoretical analysis of the proposed MUSE-TFV. Subsequently, we show the performance efficiency of MUSE-TFV with a detailed empirical evaluation.

5.1. Theoretical Analysis. We highlight the significant characteristics of MUSE-TFV in comparison with the existing multiuser searchable schemes [9, 17–19] and conjunctive search schemes [21–23] in Table 1. As the other multiuser searchable schemes [10, 11, 13, 16] utilize inverted index search structure (in inverted index based Searchable Encryption, a single common index (list of keywords) is defined for the entire set of encrypted documents), their comparison with the simple index based MUSE-TFV (in simple index searchable scheme, a separate index of keywords is associated with each encrypted document) is inapplicable here.

From Table 1, we observe that no scheme amongst the listed multiuser schemes provides a secure channel-free architecture for a token transmission. On the other hand, a conjunctive search scheme discussed in [22] offers such architecture, but it does not support multiple users in the system. In contrast, the proposed MUSE-TFV provides a conjunctive keyword based search with secure channel-free token transmission in multiuser settings. Additionally, MUSE-TFV has provision to verify the freshness of token to prevent token replay attack.

Table 2: Comparative analysis: storage-computational complexity.

Schemes	Storage overhead		Computational overhead		
	Ciphertext	Token	Encryption()	TokGen()	Search()
Hwang and Lee 2007 [9]	$(n+u)G_1$	$3G_1$	$(2+2n+u)E+P$	$2tM'+3E$	$tM'+3P$
Kiayias et al. 2016 [17]	$10G_1$	$5G_1$	$22E+(4+2u_k)M'+2P$	$8M'+13E$	$(1+u_k)M'+3P$
Zhang et al. 2016 [18]	nH	G_1	$2E+(1+n)P$	$1E$	$E+2P+U$
Wang et al. 2016 [19]	$(q+C)n$	$1H$	$2nE+2nP$	$1E$	$1P+nD$
B. Zhang and F. Zhang 2011 [21]	$(1+n)G_1+G_2$	$(2+t)G_1$	$(2+2n)E+P$	$(2+3t)E$	$(1+2t)P$
Ding et al. 2012 [22]	$(1+n)G_1$	G_1	$(2+2n)M$	$M+E$	$3M+2P$
Chen et al. 2012 [23]	$5G_1$	$4G_2$	$(4+n)E$	$4E$	$4P$
MUSE-TFV	$(\mathbf{1+n})\mathbf{G_1}$	$\mathbf{3G_1+V}$	$(\mathbf{2+2n})\mathbf{M}$	$\mathbf{6M}$	$\mathbf{2M+2P}$

n: number of keywords in the system, **t**: number of keywords in a query, **u**: number of users in the system, $\mathbf{u_k}$: number of users accessing an associated file, **H**: size of a message digest output by the used hash function, $(\mathbf{G_1}, \mathbf{G_2})$: size of an element from bilinear groups G_1 and G_2, **V**: ciphertext size of the used encryption routine, **q**: size of a random integer, **P**: pairing, **E**: exponentiation, **M**: scalar multiplication, $\mathbf{M'}$: modular multiplication, **D**: data comparison, and **U**: set union operation.

We compare the performance of MUSE-TFV with the existing schemes in terms of the storage overhead (i.e., size of a ciphertext (excluding payload) and size of a token) and computational overhead (for the proposed *Encryption()*, *TokGen()*, and *Search()* algorithms) in Table 2.

5.1.1. Storage Complexity. To show the storage overhead, we present the ciphertext/token size in terms of the size of an element from the bilinear groups (G_1, G_2). Observing Table 2, we say that the constructions given in [17, 23] are storage efficient with the constant ciphertext and token size (i.e., $O(1)$). In contrast, the proposed MUSE-TFV has a ciphertext size linear to the number of keywords in the system (i.e., $(O(n))$ that is same as ciphertext storage complexity of the existing schemes [18, 19, 21, 22].

The significant characteristic of MUSE-TFV is its constant (i.e., $O(1)$) token storage complexity. This constant overhead makes the proposed scheme as efficient as the existing schemes [9, 17, 18, 22]. In fact, the actual token size for the MUSE-TFV is three times higher than the token constructed by the schemes [18, 22]. However, with such increased token size, we offer a secure token transmission over any public channel without channel setup overhead. Moreover, with an added component V to the token (where V is the size of a ciphertext for an encrypted verification key $TVK1$), we prevent the token replay attack.

5.1.2. Computational Complexity. We present the computational overhead in terms of the major operations, namely, modular multiplication (M'), scalar multiplication (M), exponentiation (E), and pairing (P) involved in the listed schemes. From our experiments, we observe that a scalar multiplication, an exponentiation, and a pairing operation are costlier (involving more CPU cycles) than a modular multiplication operation. Therefore, from Table 2, we say that the computational cost of the proposed *Encryption()* algorithm (i.e., $(2+2n)M$) is almost same as the encryption cost of the listed multikeyword schemes [9, 21, 22]. We note that this encryption overhead is double as compared to the encryption overhead involved in the schemes [18, 23].

On the other hand, similar to the scheme in [18], MUSE-TFV has a constant computational complexity, i.e., $(O(n))$ (independent of the number of users u), for *Encryption()* algorithm. Such computational cost is far more better than the existing schemes [9, 17] with $O(n+u)$ and $O(u_k)$ encryption overhead, respectively. Therefore, we say that with moderate computational overhead for the proposed *Encryption()* algorithm MUSE-TFV supports multiple keyword based search as well as multiple users in the system.

From Table 2, we observe that the computational complexity of the proposed *TokGen()* algorithm of MUSE-TFV is same as the token construction cost of the existing schemes [18, 22, 23], i.e., $O(1)$. With such constant computational overhead, MUSE-TFV performs better than the existing schemes [9, 21] having $O(t)$ token construction overhead. Additionally, we note that *TokGen()* algorithm of MUSE-TFV consumes more CPU cycles as compared to the Token Generation algorithm of the schemes [18, 22, 23] due to its interactive token construction steps. However, with such added overhead, MUSE-TFV supports multiple users in the system.

We also note that the computational cost of a *Search()* algorithm of MUSE-TFV (i.e., $(2M+2P)$) is almost same as the existing schemes [18, 22, 23]. This constant search complexity (i.e., $O(1)$) is better than the search complexity (i.e., $O(t)$) involved in [9, 21]. Moreover, as a multiuser scheme, the MUSE-TFV offers constant computational cost $O(1)$ (i.e., independent from u) during search phase. This cost is much more better than the search computational overhead (i.e., $O(u_k)$) involved in the scheme [17]. It is worth noting that, with similar search complexity as the existing schemes [18, 22, 23], the proposed MUSE-TFV provides an additional token freshness verification feature.

5.1.3. Communication Complexity. In Table 3, we present the communication complexity of the proposed MUSE-TFV during *Data Upload*, *Token Generation*, and *Search* phases, as compared to the existing multiuser schemes [9, 17–19]. We note that, with S as a message, a scheme in [19] suffers with the highest communication overhead (i.e., $3cS$ for c

Table 3: Comparative analysis: communication complexity.

Schemes	Data Upload	Token Generation	Search
Hwang and Lee 2007 [9]	cS	—	$(1 + c)S$
Kiayias et al. 2016 [17]	cS	—	$(3 + c)S$
Zhang et al. 2016 [18]	cS	—	$2S$
Wang et al. 2016 [19]	$3cS$	—	$(1 + c)S$
MUSE-TFV	**cS**	**2qS**	**(1 + c)S**

S: a message, **c**: number of ciphertexts available at the server, and **q**: number of queries.

ciphertexts) during *Data Upload* phase wherein uploading of a single ciphertext involves three messages (i.e., a preindex message from a Data Owner to the server, an index parameter message from the server to the Data Owner, and a ciphertext message from the Data Owner to the server). In contrast, the proposed MUSE-TFV has an optimal communication overhead of a single message per ciphertext (i.e., cS messages for c ciphertexts) from a Data Owner to the server. With such overhead, the proposed scheme performs similar to the existing schemes discussed in [9, 17, 18].

A scheme in [17] uses two servers (S_{main} and S_{aid}) to perform a search operation where a communication overhead is $(3 + c)S$ messages (i.e., a token message from a user to S_{main}, a token message from a user to S_{aid}, an additional message from S_{aid} to S_{main}, and c result messages from S_{main} to the requesting user). In contrast, the proposed MUSE-TFV involves $(1+c)S$ messages (i.e., a token message from a user to the Storage Server and c result messages from the server to the user) during *Search* phase. The scheme of [18] has the lowest communication overhead during search operation, that is, $2S$ (a token message from a user to the server and a result message from the server to the user). However, in the scheme [18], the server suffers with the additional computational overhead (for set union operations) in order to incorporate c result messages into a single message.

Table 3 shows that the *Token Generation* phase of the proposed MUSE-TFV suffers with the communication overhead of $2qS$ for q queries. This overhead is due to the interactive Token Generation algorithm that involves two message exchanges between a DU and the ETA, that is, a Token Request message from a DU and a response message from the ETA. However, with such added communication overhead, we achieve a more secure system wherein every Token Generation activity is logged at the trusted site and thus any dishonest activity from a DU can easily be tracked. Moreover, with such interactive Token Generation algorithm, the proposed scheme provides a token freshness verification to prevent a token replay attack. Thus, MUSE-TFV is indeed an effective multiuser scheme for the applications where security of each search activity is a prime requirement.

5.2. Empirical Evaluation. To evaluate the performance, we conduct the experiments on 32-bit, 2.10 GHz Pentium Core 2 Duo CPU with Windows 7 machine using Java Pairing based Cryptographic (JPBC) Library [42]. From JPBC Library, we utilize Type A pairing (i.e., $G_1 \times G_1 \rightarrow G_2$) which is based on an elliptic curve $E(F_q) : y^2 = x^3 + x$. Here, the group G_1 is a subgroup of $E(F_q)$, and the cyclic group G_2 is a subgroup of $E(F_q)^2$ where q is a large prime number. The group order of G_1 is 160 bits, and the base field is 512 bits.

Table 4: Simulation parameters.

Parameters	Values for simulation
n	{50, 100, 150, 200, 250, 300}
u	{1000, 2000, 3000, 4000, 5000}
t	{10, 20, 30, 40, 50}

To systematically compare the performance of the MUSE-TFV with other schemes, we consider three significant parameters, that is, (i) number of keywords in the system (n), (ii) number of keywords in a query (t), and (iii) number of users in the system (u) (Table 4). We perform experiments for different size systems with $n \in \{50, 100, 150, 200, 250, 300\}$. For each system, we simulate the *Encryption()*, *TokGen(),* and *Search()* algorithms multiple times and consider their average results. To show the efficiency of MUSE-TFV as a multiuser scheme, we consider a different number of users, that is, $u \in \{1000, 2000, 3000, 4000, 5000\}$ in the system. Additionally, during Token Generation experiments, we select the conjunctive queries with the variable number of keywords, that is, $t \in \{10, 20, 30, 40, 50\}$. As a large number of keywords in conjunction make a query complex and impractical, we select comparatively small values for t.

From Table 2, we identify that the computational cost of *Encryption()* algorithms for all multikeyword schemes *(MKC)* [21–23] depends upon n whereas for all multiuser schemes [9, 17, 18], it depends upon n, or u or u_k. Thus, we simulate *Encryption()* algorithms for all the listed schemes with different values of n and u separately and show their responses in Figures 5(a) and 5(b), respectively. Note that for simulation purpose we consider the worst case scenario for a scheme [17], where $u_k = u$.

From the results in Figure 5(a), we note that the encryption time of the proposed MUSE-TFV is linearly increasing with the number of keywords (i.e., n). However, this time overhead is same as the encryption time overhead of [9, 21, 22] but larger than the overhead involved in [18, 23]. Additionally, from Figure 5(b), we observe that the existence of multiple users in the system does not affect the time consumption of encryption algorithm of MUSE-TFV. This characteristic makes the MUSE-TFV more practical than the existing multiuser schemes [9, 17] where the encryption time overhead is linearly increasing with the number of users. Here, we say that with the constant encryption overhead (i.e., independent of the number of users (u)) the *Encryption()* algorithm of MUSE-TFV supports multiple keywords in a ciphertext and multiple users in the system.

We present the empirical results for *TokGen()* algorithm of MUSE-TFV and other multikeyword (MKQ) schemes in Figure 6. From these results, we say that the MUSE-TFV takes almost constant time to construct a token regardless of the number of keywords in a query. With this characteristic, MUSE-TFV resembles the schemes [22, 23] and performs

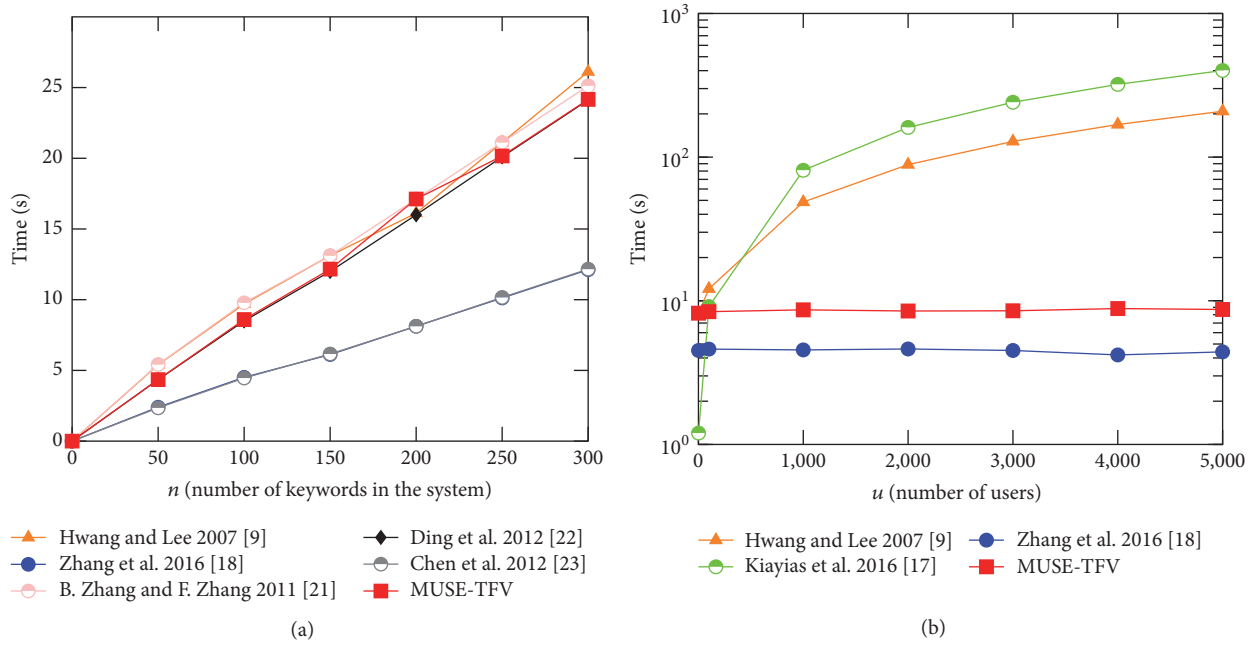

Figure 5: Simulation results for *Encryption()* algorithm.

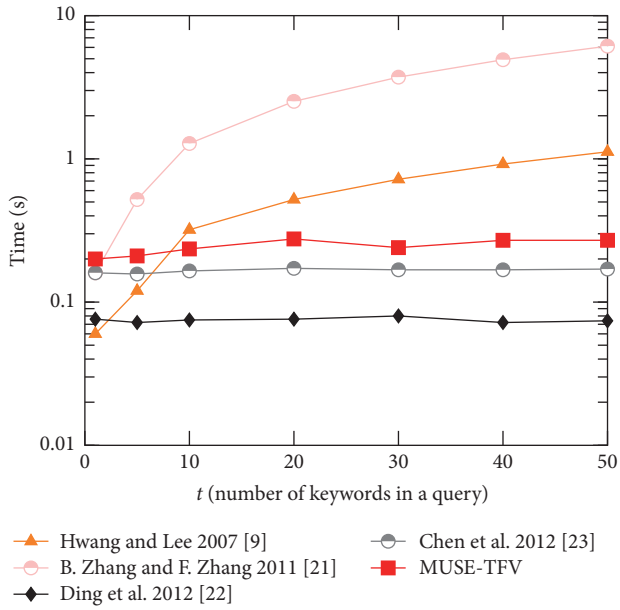

Figure 6: Simulation results for *TokGen()* algorithm.

better than the other multikeyword schemes [9, 21] having $O(t)$ token computational overhead. However, MUSE-TFV takes more time as compared to [22, 23] because of its interactive nature.

According to Table 2, the computational overhead for the *Search()* algorithm of the listed schemes is either constant or otherwise depending upon t or u_k. Thus, we simulate the listed schemes for their *Search()* algorithm with different values of t and u separately and show their responses in Figures 7(a) and 7(b), respectively.

Observing the results in Figure 7(a), we note that the search time overhead for MUSE-TFV is almost constant and independent of the number of keywords in a query (t). With this characteristic, the MUSE-TFV performs a conjunctive search with much less computational time as compared to the existing conjunctive search schemes [9, 21] where the search time is affected by the number of keywords in query (t). From the results in Figure 7(b), we note that, with constant search time overhead, the proposed MUSE-TFV supports multiple users in the system as efficiently as the scheme [18]. In addition, we say that with the search time linear to the number of users (u) the scheme of [17] is indeed less practical. In contrast, with the constant search time overhead, the MUSE-TFV performs a conjunctive keyword search in response to a query coming from any user in the multiuser settings.

At last we claim that our empirical results are completely in accordance with the theoretically measured computational complexity presented in Table 2. From the theoretical analysis and empirical evaluation, we conclude that, with the moderate storage-computational overhead, the proposed MUSE-TFV is an elegant multiuser searchable scheme with a provision of conjunctive keyword search and token freshness verification.

6. Concluding Remarks

In this paper, we discuss the proposed MUSE-TFV: a Multiuser Searchable Encryption with Token Freshness Verification that is based on the concept of Functional Encryption. Unlike the existing Functional Encryption based multiuser searchable schemes wherein a user generates a search token using his own search key, in the proposed MUSE-TFV, a Data User, DU, constructs a search token in cooperation with the

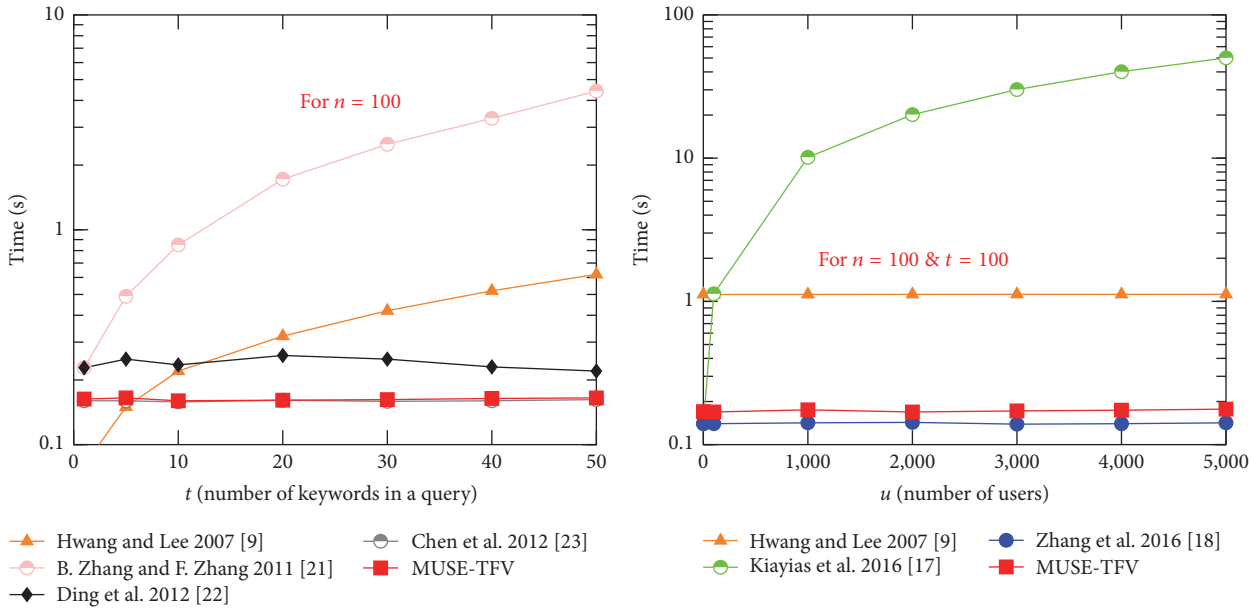

Figure 7: Simulation results for *Search()* algorithm.

ETA. With such interactive Token Generation mechanism, every search activity of each DU is logged at the enterprise trusted site and thus dishonest activity can be easily captured. Moreover, in the MUSE-TFV, each constructed token is valid for one-time use and its freshness is checked at the SS using a verification key issued by the ETA. Such token verification procedure prevents the reuse of the same token and so the MUSE-TFV avoids token replay attack. Additionally, we provide a secure channel-free token transmission as well as a conjunctive keyword search with the proposed scheme.

With a security analysis, we prove the correctness of the proposed MUSE-TFV against chosen keyword attack and token replay attack. With a detailed theoretical analysis, we justify the efficiency of the proposed scheme. Additionally, we evaluate the performance of the proposed scheme based on three significant parameters: number of users, number of keywords in the system, and the number of keywords in conjunctive query. Our experimental evaluation shows that, with almost same computational-storage overhead as the existing conjunctive keyword search schemes, the proposed MUSE-TFV provides the additional features of multiuser support and token freshness verification.

Notations

pp:	System's public parameters
msk:	Master secret key
n:	Number of keywords in the system
u:	Number of users in the system
$\mathcal{KS}$:	Keyword space that involves n keywords
(Y, y):	Server's public-private key pair
(X, x):	User's public-private key pair
$M' = E_{key}(M)$:	Encrypted payload message where E is any symmetric key cipher with a key key
$W = \{w_j \mid 1 \le j \le n\}$:	A list of n keywords associated with a ciphertext
C:	A ciphertext
PT:	A partial token
T:	A token
t:	Number of keywords in a conjunctive query
$Q' = \{W', I'\}$:	A conjunctive query that involves two sets (W', I') where $W' = \{w'_j\}_{j \in I'}$ is a list of t keywords and $I' = \{\ell_1, \ell_2, \ldots, \ell_t\}$ is a list of positions of keywords in $\mathcal{KS}$
C_i:	ith ciphertext
$(TVK1, TVK2)$:	Token verification keys
R:	A search result.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Abdalla, M. Bellare, D. Catalano et al., "Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions," in *Advances in cryptology—CRYPTO 2005*, vol. 3621 of *Lecture Notes in Comput. Sci.*, pp. 205–222, Springer, Berlin, 2005.

[2] S. Shin and K. Kobara, "Towards secure cloud storage," *Demo for CloudCom 2010*, vol. 2, p. 8, 2010.

[3] G. Brunette, R. Mogull, and et al., "Security guidance for critical areas of focus in cloud computing v2. 1," *Cloud Security Alliance*, pp. 1–76, 2009.

[4] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the in Security and Privacy, 2000 SP2000 IEEE Symposiumon. 1em plus 0.5em minus 0.4em IEEE*, pp. 44–55, 2000.

[5] E.-J. Goh, "Secure indexes," *IACR Cryptology ePrint Archive*, vol. 2003, p. 216, 2003.

[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology—EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 506–522, Springer, Berlin, Germany, 2004.

[7] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings in Computational Science and Its Applications–ICCSA International Conference, Part I 2008. 1em plus 0.5em minus 0.4em*, vol. 30, pp. 1249–1259, Springer, Perugia, Italy, 2008.

[8] D. J. Park, K. Kim, and P. J. Lee, "Public Key Encryption with Conjunctive Field Keyword Search," in *Information Security Applications*, vol. 3325 of *Lecture Notes in Computer Science*, pp. 73–86, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[9] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Pairing-based cryptography—Pairing 2007*, vol. 4575 of *Lecture Notes in Comput. Sci.*, pp. 2–22, Springer, Berlin, 2007.

[10] P. Wang, H. Wang, and J. Pieprzyk, "Common secure index for conjunctive keyword-based retrieval over encrypted data," *Secure Data Management*, pp. 108–123, 2007.

[11] P. Wang, H. Wang, and J. Pieprzyk, "Threshold privacy preserving keyword searches," in *Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science. 1em plus 0.5em minus 0.4em , 2008, pp. 646–658*, pp. 646–658, Springer, 2008.

[12] P. Wang, H. Wang, and J. Pieprzyk, "An Efficient Scheme of Common Secure Indices for Conjunctive Keyword-Based Retrieval on Encrypted Data," in *Information Security Applications*, vol. 5379, pp. 145–159, Springer, Berlin, Heidelberg, 2009.

[13] P. Wang, H. Wang, and J. Pieprzyk, "Keyword Field-Free Conjunctive Keyword Searches on Encrypted Data and Extension for Dynamic Groups," in *Cryptology and Network Security*, vol. 5339, pp. 178–195, Springer, Berlin, Heidelberg, 2008.

[14] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Information security practice and experience*, vol. 4991 of *Lecture Notes in Comput. Sci.*, pp. 71–85, Springer, Berlin, 2008.

[15] J. Li and X. Chen, "Efficient multi-user keyword search over encrypted data in cloud computing," *Computing and Informatics*, vol. 32, no. 4, pp. 723–738, 2013.

[16] H. Huang, J. Du, H. Wang, and R. Wang, "A multi-keyword multi-user searchable encryption scheme based on cloud storage," in *Proceedings of the Joint 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 10th IEEE International Conference on Big Data Science and Engineering and 14th IEEE International Symposium on Parallel and Distributed Processing with Applications, IEEE TrustCom/BigDataSE/ISPA 2016*, pp. 1937–1943, August 2016.

[17] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, in *Proceedings of the European Symposium on Research in Computer Security. 1em plus 0.5em minus 0.4em*, pp. 173–195, Springer, 2016.

[18] Y. Zhang, L. Liu, and S. Wang, "Multi-User and Keyword-Based Searchable Encryption Scheme," in *Proceedings of the 2016 12th International Conference on Computational Intelligence and Security (CIS)*, pp. 223–227, Wuxi, China, December 2016.

[19] S. Wang, X. Zhang, and Y. Zhang, "Efficiently multi-user searchable encryption scheme with attribute revocation and grant for cloud storage," *PLoS ONE*, vol. 11, no. 11, Article ID e0167157, 2016.

[20] J. Ye, J. Wang, J. Zhao, J. Shen, and K.-C. Li, "Fine-grained searchable encryption in multi-user setting," *Soft Computing*, pp. 1–12, 2016.

[21] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 262–267, 2011.

[22] M. Ding, F. Gao, Z. Jin, and H. Zhang, "An efficient public key encryption with conjunctive keyword search scheme based on pairings," *Proceedings in 2012 3rd IEEE International Conference on Network Infrastructure and Digital Content. 1em plus 0.5em minus 0.4em IEEE*, pp. 526–530, 2012.

[23] Z. Chen, C. Wu, and D. Wang, "Conjunctive keywords searchable encryption with efficient pairing, constant ciphertext and short trapdoor," *PAISI*, pp. 176–189, 2012.

[24] Y.-C. Chang and M. Mitzenmacher, "Privacy Preserving Keyword Searches on Remote Encrypted Data," in *Applied Cryptography and Network Security. 1em plus 0.5em minus 0.4em*, vol. 3531, pp. 442–455, Springer, Berlin, Heidelberg, 2005.

[25] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[26] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security: Second International Conference, ACNS 2004, Yellow Mountain, China, June 8–11, 2004. Proceedings*, vol. 3089 of *Lecture Notes in Computer Science*, pp. 31–45, Springer, Berlin, Germany, 2004.

[27] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 3783, pp. 414–426, 2005.

[28] J. W. Byun, D. H. Lee, and J. Lim, "Efficient Conjunctive Keyword Search on Encrypted Data Storage System," in *Public Key Infrastructure. 1em plus 0.5em minus 0.4em*, vol. 4043, pp. 184–196, Springer, Berlin, Heidelberg, 2006.

[29] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography Conference: TCC 2007. 1em plus 0.5em minus 0.4em*, pp. 535–554, Springer, Berlin, Germany, 2007.

[30] M.-S. Hwang, S.-T. Hsu, and C.-C. Lee, "A new public key encryption with conjunctive field keyword search scheme," *Information Technology and Control*, vol. 43, no. 3, pp. 277–288, 2014.

[31] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *The Journal of Systems and Software*, vol. 83, no. 5, pp. 763–771, 2010.

[32] Y. Zhao, H. Ma, X. Chen, Q. Tang, and H. Zhu, "A new trapdoor-indistinguishable public key encryption with keyword search," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 3, no. 1-2, pp. 72–81, 2012.

[33] Y. Miao, J. Ma, F. Wei, Z. Liu, X. A. Wang, and C. Lu, "VCSE: Verifiable conjunctive keywords search over encrypted data without secure-channel," *Peer-to-Peer Networking and Applications*, vol. 10, no. 4, pp. 995–1007, 2017.

[34] N. Haller, "The S/KEY One-Time Password System," RFC Editor RFC1760, 1995.

[35] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, 2001.

[36] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology: Proceedings of (CRYPTO '84)*, vol. 196 of *Lecture Notes in Computer Science*, pp. 47–53, Springer, Berlin, Germany, 1985.

[37] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, May 2007.

[38] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 89–98, November 2006.

[39] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology – EUROCRYPT 2005*, vol. 3494 of *Lecture Notes in Computer Science*, pp. 457–473, Springer, Berlin, Germany, 2005.

[40] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Advances in Cryptology—EUROCRYPT 2008*, vol. 4965 of *Lecture Notes in Computer Science*, pp. 146–162, Springer, Berlin, Germany, 2008.

[41] C.-C. Lee, S.-T. Hsu, and M.-S. Hwang, "A study of conjunctive keyword searchable schemes," *IJ Network Security*, vol. 15, no. 5, pp. 321–330, 2013.

[42] A. de Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications (ISCC '11)*, pp. 850–855, July 2011.