Expressive and Secure Searchable Encryption in the Public Key Setting

Zhiquan Lv[1,2], Cheng Hong[1], Min Zhang[1], and Dengguo Feng[1]

[1] Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
{lvzhiquan,hongcheng,mzhang,feng}@tca.iscas.ac.cn

Abstract. Searchable encryption allows an untrusted server to search on encrypted data without knowing the underlying data contents. Traditional searchable encryption schemes focus only on single keyword or conjunctive keyword search. Several solutions have been recently proposed to design more expressive search criteria, but most of them are in the setting of symmetric key encryption. In this paper, based on the composite-order groups, we present an expressive and secure asymmetric searchable encryption (ESASE) scheme, which is the first that simultaneously supports conjunctive, disjunctive and negation search operations. We analyze the efficiency of ESASE and prove it is secure under the standard model. In addition, we show that how ESASE could be extended to support the range search and the multi-user setting.

Keywords: Searchable Encryption, Asymmetric Searchable Encryption, Expressive Search

1 Introduction

In a remote storage system, a user can store his data in the remote server and then visit the data using his PC or mobile devices. Since the server cannot always be fully trusted, data containing sensitive information must be encrypted to protect the user's privacy. However, it makes retrieval of such encrypted data difficult. To cope with this problem, searchable encryption schemes have been proposed, which can be divided into two versions: symmetric searchable encryption (SSE) and asymmetric searchable encryption (ASE). Generally speaking, SSE is more efficient than ASE, while ASE is more suitable to practical scenarios since it does not require that different users must share a common secret key.

However, most traditional searchable encryption schemes focus only on single keyword search [1–3, 5, 17] or multiple keyword search [4, 8, 9, 15, 16], practical systems desire a more expressive search. In the symmetric key setting, some solutions have been recently designed for general boolean queries on encrypted data [7, 18]. However, in the public key setting, to the best of our knowledge, there are only two related works [6, 12].

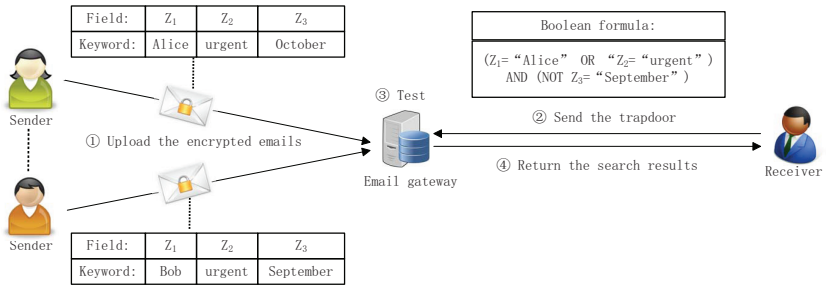

Fig. 1: An example of secure searchable email system.

Consider a secure searchable email system in Figure 1. To support the keyword search, each email will be defined some keyword fields, such as "*Sender*", "*Priority*", and "*Month*". Here we use "Z_1", "Z_2" and "Z_3" to denote these fields respectively. Before sending an email, a sender first encrypts the email content using a standard public key encryption algorithm with the receiver's public key, and then appends some additional encrypted keywords of the keyword fields, such as "*Alice*", "*urgent*", and "*October*". Generally speaking, the ASE schemes focus only on the keywords encryption since it is well known that the standard public key encryption is secure. In schemes [6, 12], an email gateway might be given a trapdoor for the boolean formula (Z_1 = "*Alice*" OR Z_2 = "*urgent*"), which indicates that the receiver wants the gateway to return all emails either sent by Alice or having urgent priority. Since the receiver might have read all the emails in September, the trapdoor for boolean formula would actually like ((Z_1 = "*Alice*" OR Z_2 = "*urgent*") AND (NOT Z_3 = "*September*")). However, neither of schemes [6, 12] can work in the situations involving negation search operation. Moreover, the former is less secure and the latter is less efficient.

The requirement of supporting negation also can be found in Attribute-Based Encryption schemes, where the solutions are polynomial interpolation [20] and "2-equation technique" [21, 22]. However, these solutions cannot be applied to searchable encryption because the keywords will be disclosed to the server.

Our Contributions. Our contributions are summarized as follows.

- Based on the composite-order groups, we propose an expressive and secure ASE scheme named ESASE. To the best of our knowledge, ESASE is the first that simultaneously supports conjunctive, disjunctive and negation search operations in the public key setting.
- We give a detail security proof of ESASE under the standard model. Compared with [6], ESASE does not disclose the searching keywords in the trapdoor. Furthermore, the efficiency analysis shows that the overhead on storage, communication, and computation in ESASE is close to [6], but much lower than [12].

– We show how ESASE could be extended to support a class of simple range search, which can help the user choose the search ranges based on different choices of granularity. Furthermore, we extend ESASE to the multi-user setting, which can minimize the overhead on computation and communication.

1.1 Related Work

Song et al. [1] initiate the research on searchable encryption and present the first practical solution in the symmetric key setting. After this work, several works [2, 3, 11, 13, 14, 17] have been proposed to improve the efficiency of the system or provide stronger security. Boneh et al. [5]. first address the concept of asymmetric searchable encryption, where anyone can use the public key to encrypt the data and keywords but only authorized users with secret key can search. However, these early schemes focus only on single keyword search.

Evidently, there are two trivial solutions to achieve conjunctive keyword search based on the single keyword search: one is to obtain the intersection of all sets where each set is the searching result for every individual keyword in the conjunction; another would be to define a meta-keyword for each possible keywords conjunction. However, as Golle et al. [4] point out, these approaches suffer from obvious drawbacks. The former allows the gateway to learn which documents contain each individual keyword based on the results of the conjunctive query, and the latter requires an exponential storage and search time for the number of keywords. Therefore, they propose the first solution for symmetric conjunctive keyword search, where each encrypted file is associated with encrypted keywords that are assigned to separate keyword fields. If a user queries a trapdoor with several keywords, the server can search a file containing those keywords with this trapdoor. Later, Park et al. [16] propose the notion of asymmetric conjunctive keyword search. To decrease the trapdoor size and the computation overhead, several solutions are presented in [8, 9, 15].

To design more expressive search criteria, Cash et al. [18] propose the first searchable symmetric encryption protocol that supports conjunctive search and general boolean queries on outsourced data. One of the drawbacks of their scheme is that the trapdoors they generate are deterministic, in the sense that the same trapdoor will always be generated for the same keyword. Thus, it leaks statistical information about the user's search pattern [19]. Based on the orthogonalization of the keyword field according to the Gram-Schmidt process, Moataz et al. [7] present a similar solution which avoids this drawback in [18]. In the public key setting, Katz et al. [12] propose an inner-product predicate encryption scheme, which can enable the disjunctive search over encrypted data using polynomial evaluation. However, as pointed out in [12], the complexity is proportional to d^t, where t is the number of variables and d is the maximum degree (of the resulting polynomial) in each variable. Recently, Lai et al. [6] present a more efficient construction based on the fully secure key-policy attribute-based encryption (KP-ABE) scheme [10]. Compared with [12], the size of a ciphertext (or a trapdoor) in [6] is linear with the number of keyword fields (or the size of the search predicate), not superpolynomial. However, scheme [6] discloses the

searching keywords in the trapdoor, which will let the server learn whether the encrypted data contains the keywords in the trapdoor.

2 PRELIMINARIES

2.1 Linear Secret Sharing Schemes

Our construction will make essential use of linear secret-sharing schemes (LSSS). We first describe the definition of access structure [23] that will be used in LSSS. Then, we give the definition of LSSS adapted from [23].

Definition 1 (Access Structure). *Let* $\{P_1, \ldots, P_m\}$ *be a set of parties. A collection* $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_m\}}$ *is monotone if* $\forall B, C$: *if* $B \in \mathbb{A}$ *and* $B \subseteq C$ *then* $C \in \mathbb{A}$. *An* access structure *(respectively, monotonic access structure) is a collection (respectively, monotone collection)* $\mathbb{A}$ *of non-empty subsets of* $\{P_1, \ldots, P_m\}$, *i.e.,* $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_m\}} \backslash \{\emptyset\}$. *The sets in* $\mathbb{A}$ *are called the* authorized sets, *and the sets not in* $\mathbb{A}$ *are called the* unauthorized sets.

Definition 2 (Linear Secret-Sharing Schemes (LSSS)). *Suppose there exists a linear secret sharing structure* $\mathbb{A} = (\mathbf{A}, \rho)$, *where* $\mathbf{A}$ *is a* $\ell \times m$ *matrix and* ρ *is an injective function from* $\{1, \ldots, \ell\}$ *to a party. Let* $S \in \mathbb{A}$ *be any authorized set, and* $I \subset \{1, \ldots, \ell\}$ *be defined as* $I = \{i : \rho(i) \in S\}$. *Therefore, there exist constants* $\{\sigma_i \in \mathbb{Z}_p\}$ *such that* $\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0)$, *where* A_i *is the i'th row of matrix* $\mathbf{A}$. *When we consider the column vector* $\upsilon = (s, r_2, \ldots, r_m)$, *where* $s \in \mathbb{Z}_p$ *is the secret to be shared, and* $r_2, \ldots, r_m \in \mathbb{Z}_p$ *are randomly chosen, then* $\mathbf{A}\upsilon$ *can be regarded as the linear secret sharing where each share* $A_i\upsilon$ *belongs to party* $\rho(i)$. *On the other hand, given a party set* S *and its corresponding rows* $I = \{i : \rho(i) \in S\}$ *in the matrix* $\mathbf{A}$, *finding* $\{\sigma_i \in \mathbb{Z}_p\}$ *such that* $\sum_{i \in I} \sigma_i A_i \upsilon = s$ *is called linear secret reconstruction. Note that, for unauthorized sets, no such constants exist.*

In our context, the role of the parties is taken by the keywords. Access structures (i.e., search predicates) might also be described in terms of monotonic boolean formulas. As described in [23], any monotonic boolean formula can be converted into an LSSS representation.

2.2 Composite-Order Bilinear Groups

We will construct our scheme in composite-order bilinear groups [24]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of order $N = p_1 p_2 p_3 p_4$, where p_1, p_2, p_3 and p_4 are distinct primes. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ be a bilinear map such that $e(g, g) \neq 1$, and for any $u, v \in \mathbb{G}$, $a, b \in \mathbb{Z}_N$ it holds $e(u^a, v^b) = e(u, v)^{ab}$. We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map e are both efficiently computable.

Let $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$, $\mathbb{G}_{p_3}$ and $\mathbb{G}_{p_4}$ denote the subgroups of order p_1, p_2, p_3 and p_4 in $\mathbb{G}$ respectively. Observe that $\mathbb{G} = \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3} \times \mathbb{G}_{p_4}$. We note that if

$h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$ for $i \neq j$, $e(h_i, h_j) = 1$. To see this, suppose $h_1 \in \mathbb{G}_{p_1}$ and $h_2 \in \mathbb{G}_{p_2}$. We let g denote a generator of $\mathbb{G}$. Then, $g^{p_2 p_3 p_4}$ generates $\mathbb{G}_{p_1}$, and $g^{p_1 p_3 p_4}$ generates $\mathbb{G}_{p_2}$. Hence, we can have:

$$e(h_1, h_2) = e((g^{p_2 p_3 p_4})^{\alpha_1}, (g^{p_1 p_3 p_4})^{\alpha_2}) = e(g^{p_3 p_4 \alpha_1}, g^{\alpha_2})^{p_1 p_2 p_3 p_4} = 1,$$

where $\alpha_1 = \log_{g^{p_2 p_3 p_4}} h_1$ and $\alpha_2 = \log_{g^{p_1 p_3 p_4}} h_2$. This orthogonality property of $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$, $\mathbb{G}_{p_3}$, $\mathbb{G}_{p_4}$ will be a principal tool in our construction and security proof.

2.3 Complexity Assumptions

We now state the complexity assumptions that will be used in our security proof. The first two assumptions are also used in [6] and the third can be easily proved by utilizing the theorems proposed in [12]. We note all of them hold in the generic group model.

Assumption 1. Let $N, \mathbb{G}, \mathbb{G}_T, e$ be defined as in above section. Let $g \in \mathbb{G}_{p_1}$, $X_3 \in \mathbb{G}_{p_3}$, and $X_4 \in \mathbb{G}_{p_4}$ be chosen at random. Given the tuple ($\mathbb{G}$, $\mathbb{G}_T$, N, e, g, X_3, X_4), this assumption states that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the random elements in $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_1 p_2}$ with non-negligible advantage.

Assumption 2. Let $N, \mathbb{G}, \mathbb{G}_T, e$ be defined as in above section. Let $g, X_1 \in \mathbb{G}_{p_1}$, $X_2, Y_2 \in \mathbb{G}_{p_2}$, $X_3, Y_3 \in \mathbb{G}_{p_3}$, and $X_4 \in \mathbb{G}_{p_4}$ be chosen at random. Given the tuple ($\mathbb{G}$, $\mathbb{G}_T$, N, e, g, $X_1 X_2$, $Y_2 Y_3$, X_3, X_4), this assumption states that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the random elements in $\mathbb{G}_{p_1 p_2 p_3}$ and $\mathbb{G}_{p_1 p_3}$ with non-negligible advantage.

Assumption 3. Let $N, \mathbb{G}, \mathbb{G}_T, e$ be defined as in above section. Let $s \in \mathbb{Z}_N$, $g, u \in \mathbb{G}_{p_1}$, $g_2, X_2 \in \mathbb{G}_{p_2}$, $X_3 \in \mathbb{G}_{p_3}$, $X_4, h' \in \mathbb{G}_{p_4}$, and $B_{24}, D_{24} \in \mathbb{G}_{p_2 p_4}$ be chosen at random. Given the tuple $(\mathbb{G}, \mathbb{G}_T, N, e, g, g_2, X_2, uh', g^s B_{24}, X_3, X_4)$, this assumption states that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish $u^s D_{24}$ and a random element in $\mathbb{G}_{p_1 p_2 p_4}$ with non-negligible advantage.

3 Syntax and Security Model

3.1 Syntax

We consider that a user' s encrypted data is outsourced in the storage of an untrusted server, such as an email gateway. To support the keyword search, we will define some keyword fields for the emails, such as "Sender", "Priority", "Month". Suppose each email is associated with a keyword set $W = \{w_1, \ldots, w_n\}$, where w_i is the keyword of the email in the i^{th} keyword field and n is the number of keyword fields. Moreover, we employ the same assumption used in the previous works [4, 6, 9, 16]: every keyword field is defined for every email. An ESASE scheme consists of the following fundamental algorithms:

Setup(1^κ): takes as input a security parameter 1^κ, and generates a public key pk and secret key sk.

Encrypt(pk, W): for the public key pk and a keyword set W, produces a ciphertext C_W.

Trapdoor(pk, sk, $\mathcal{P}$): given the public key pk, the secret key sk, and a predicate $\mathcal{P}$, produces a trapdoor $T_\mathcal{P}$.

Test(pk, C_W, $T_\mathcal{P}$): takes as input the public key pk, the ciphertext C_W, and the trapdoor $T_\mathcal{P}$. It outputs "1" if the keyword set W satisfies the predicate $\mathcal{P}$ and "0" otherwise.

3.2 Security Model

In ESASE, we assume the server who runs the **Test** algorithm is Honest but Curious. That is to say, it will execute correctly the proposed algorithm, but try to find out as much secret information as possible based on its inputs. Moreover, we define the security notion in the sense of indistinguishability security against chosen predicate attacks. Formally, security is defined using the following game between an attacker and a challenger.

Setup. The challenger takes a security parameter 1^κ and runs the Setup algorithm. The public key pk is given to the adversary. The challenger keeps the secret key sk to itself.

Phase 1. The adversary adaptively queries a number of predicates, $\mathcal{P}_1, \ldots, \mathcal{P}_q$, to the challenger. In response, the challenger runs the Trapdoor algorithm and gives the trapdoor $T_{\mathcal{P}_i}$ to the adversary, for $1 \leq i \leq q$.

Challenge. The adversary selects two target keyword sets W_0, W_1, and sends them to the challenger. The challenger picks a random bit $\beta \in \{0, 1\}$ and obtains the ciphertext C_{W_β} by running the Encrypt algorithm. Then, he gives C_{W_β} to the adversary. Note that W_0 and W_1 cannot satisfy any of queried predicates in phase 1.

Phase 2. The adversary additionally queries the challenger for trapdoors corresponding to predicates with the restriction that none of them can be satisfied by W_0 and W_1.

Guess. The adversary outputs a guess β' of β and wins the game if $\beta' = \beta$.

The advantage of the adversary in this game is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 3. *An ESASE scheme is secure if all polynomial time adversaries have at most a negligible advantage in this security game.*

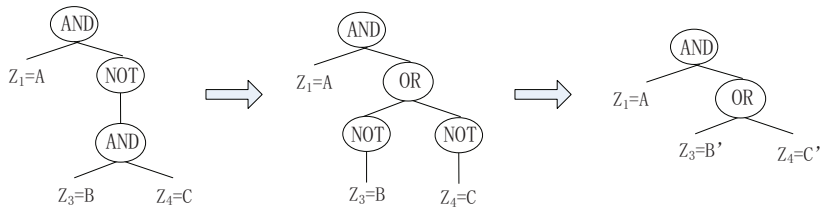

Fig. 2: An example of propagating the negation operations.

4 Construction

4.1 Overview

Consider any search predicate with negation operations, we can use the De-Morgan's law to propagate the negation operations. For example, in Figure 2, a search predicate (Z_1 = "A" AND (NOT (Z_3 = "B" AND Z_4 = "C"))) equals (Z_1 = "A" AND ((NOT Z_3 = "B") OR (NOT Z_4 = "C"))), where A, B and C denote the keywords in keyword fields Z_1, Z_3 and Z_4 respectively. After the propagation, the negation operations can only be associated with keywords. Thus, the name of the keywords in our scheme may be of two types: either the name is normal (like x) or it is *primed* (like x'). We conceptually associate primed keywords as representing the negation of unprimed keywords. Therefore, the search predicate (Z_1 = "A" AND (NOT (Z_3 = "B" AND Z_4 = "C"))) can be parsed as (Z_1 = "A" AND (Z_3 = "B'" OR Z_4 = "C'")).

As described previously, any monotonic boolean search predicate can be converted into an LSSS representation $\mathbb{A} = (\mathbf{A}, \rho)$, where $\mathbf{A}$ is a $\ell \times m$ matrix and ρ is a map from each row of $\mathbf{A}$ to a keyword field (i.e., ρ is a function from $\{1, \ldots, \ell\}$ to $\{1, \ldots, n\}$). Let $\mathcal{E} = \{t_{\rho(1)}, \ldots, t_{\rho(\ell)}\}$, $\mathcal{F} = \{f_{\rho(1)}, \ldots, f_{\rho(\ell)}\}$, where $t_{\rho(i)}$ is the keyword of keyword field $\rho(i)$ specified by the predicate and $f_{\rho(i)}$ denotes whether the keyword of keyword field $\rho(i)$ is primed or not (We can use "$-$" and "$+$" to denote it is primed and not primed respectively).

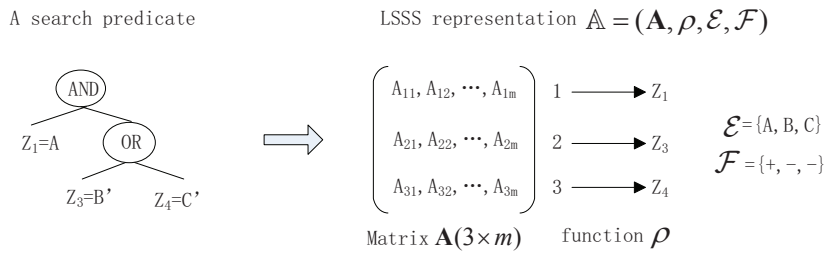

Fig. 3: An example of the conversion.

Thus, in our scheme, any boolean search predicate, which simultaneously supports the operations of conjunctive and disjunctive and negation, can be converted into an LSSS representation $\mathbb{A} = (\mathbf{A}, \rho, \mathcal{E}, \mathcal{F})$. A keyword set $W = \{w_1, \ldots, w_n\}$ satisfies a predicate $\mathbb{A} = (\mathbf{A}, \rho, \mathcal{E}, \mathcal{F})$ if and only if there exist $I \subset \{1, \ldots, \ell\}$ and constants $\{\sigma_i\}_{i \in I}$ such that $\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0)$, where A_i is the i'th row of matrix $\mathbf{A}$. Note that for $\forall i \in I$, if $f_{\rho(i)}$ denotes the keyword of keyword field $\rho(i)$ is not primed, then $t_{\rho(i)} = w_{\rho(i)}$; else, $t_{\rho(i)} \neq w_{\rho(i)}$. In addition, we define $I_{\mathbf{A},\rho}$ as the set of minimum subsets of $\{1, \ldots, \ell\}$ that satisfies $(\mathbf{A}, \rho, \mathcal{E}, \mathcal{F})$. Figure 3 shows an example of converting a search predicate into an LSSS representation, where $\ell = 3$ and $I_{\mathbf{A},\rho} = \{\{1, 2\}, \{1, 3\}\}$.

4.2 Main Construction

Setup(1^κ): Take as input a security parameter 1^κ, it returns $\texttt{params} = (p_1, p_2, p_3, p_4, \mathbb{G}, \mathbb{G}_T, e)$ with $\mathbb{G} = \mathbb{G}_{p_1} \times \mathbb{G}_{p_2} \times \mathbb{G}_{p_3} \times \mathbb{G}_{p_4}$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order $N = p_1 p_2 p_3 p_4$. Next it chooses $g, u_1, \ldots, u_n \in \mathbb{G}_{p_1}$, $X_3 \in \mathbb{G}_{p_3}$, $X_4, h_1, \ldots, h_n \in \mathbb{G}_{p_4}$ and $\alpha \in \mathbb{Z}_N$ uniformly at random. The public key pk is published as:

$$N, g, g^\alpha, \{H_i = u_i \cdot h_i\}_{1 \leq i \leq n}, X_4.$$

The secret key $sk = \{u_1, \ldots, u_n, \alpha, X_3\}$.

Encrypt(pk, W): The encryption algorithm chooses $s \in \mathbb{Z}_\mathbb{N}$ and $h, Z_0, \{Z_{1,i}\}_{1 \leq i \leq n} \in \mathbb{G}_{p_4}$ uniformly at random. Let the keyword set be $W = (w_1, \ldots, w_n) \in \mathbb{Z}_N^n$, then the corresponding ciphertext $C_W = (\widetilde{C}, C_0, \{C_i\}_{1 \leq i \leq n})$ is computed as:

$$\widetilde{C} = e(g, g^\alpha)^s, C_0 = (gh)^s \cdot Z_0, C_i = (H_i{}^{w_i})^s \cdot Z_{1,i}.$$

Trapdoor$(pk, sk, \mathcal{P})$: Suppose that the predicate $\mathcal{P}$ is corresponding to $(\mathbf{A}, \rho, \mathcal{E}, \mathcal{F})$, where $\mathbf{A}$ is the $\ell \times m$ matrix, $\rho(i)$ is a map from each row A_i of $\mathbf{A}$ to $\{1, \ldots, n\}$, $\mathcal{E} = \{t_{\rho(1)}, \ldots, t_{\rho(\ell)}\} \in \mathbb{Z}_N^\ell$, and $\mathcal{F} = \{f_{\rho(1)}, \ldots, f_{\rho(\ell)}\}$. The algorithm first chooses a random vector $\upsilon \in \mathbb{Z}_N^m$ such that $\mathbf{1} \cdot \upsilon = \alpha$, where $\mathbf{1} = (1, 0, \ldots, 0)$. Then, for each row A_i of $\mathbf{A}$, it chooses a random $r_i \in \mathbb{Z}_N$ and random elements $V_{1,i}, V_{2,i}, V_{3,i} \in \mathbb{G}_{p_3}$. The trapdoor key $T_\mathcal{P} = ((\mathbf{A}, \rho, \mathcal{F}), \{D_{1,i}, D_{2,i}, D_{3,i}\}_{1 \leq i \leq \ell})$ is computed as

$$D_{1,i} = g^{A_i \cdot \upsilon} (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i}, D_{2,i} = g^{r_i} \cdot V_{2,i}, D_{3,i} = (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{3,i}.$$

Test$(pk, C_W, T_\mathcal{P})$: Given a ciphertext C_W and a trapdoor $T_\mathcal{P}$, the algorithm first calculates $I_{\mathbf{A},\rho}$. Then, it checks if there exists an set $I \in I_{\mathbf{A},\rho}$ that satisfies

$$\widetilde{C} = \prod_{i \in I} (U_i)^{\sigma_i},$$

where $\sum_{i \in I} \sigma_i A_i = (1, 0, \ldots, 0)$. For $i \in I$: if $f_{\rho(i)}$ denotes the keyword of keyword field $\rho(i)$ is not primed, the above $U_i = e(D_{1,i}, C_0)/e(D_{2,i}, C_{\rho(i)})$; else, $U_i = e(D_{1,i}, C_0)/e(D_{3,i}, C_0)$ and the equality $e(C_0, D_{3,i}) = e(C_{\rho(i)}, D_{2,i})$ does not hold.

If no element in $I_{\mathbf{A},\rho}$ satisfies it, it outputs "0". Otherwise, it outputs "1".

4.3 Correctness

In the Test algorithm, for $i \in I$ such that $f_{\rho(i)}$ denotes the keyword of keyword field $\rho(i)$ is not primed, if $t_{\rho(i)}$ is equal to $w_{\rho(i)}$, we can have:

$$\begin{aligned} U_i &= e(D_{1,i}, C_0)/e(D_{2,i}, C_{\rho(i)}) \\ &= \frac{e(g^{A_i \cdot \upsilon}(u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i}, (gh)^s \cdot Z_0)}{e(g^{r_i} \cdot V_{2,i}, (H_{\rho(i)}{}^{w_{\rho(i)}})^s \cdot Z_{1,\rho(i)})} \\ &= \frac{e(g^{A_i \cdot \upsilon}(u_{\rho(i)})^{t_{\rho(i)} \cdot r_i}, g^s)}{e(g^{r_i}, u_{\rho(i)}{}^{w_{\rho(i)} s})} \\ &= e(g,g)^{sA_i \cdot \upsilon}. \end{aligned}$$

For $i \in I$ such that $f_{\rho(i)}$ denotes the keyword of keyword field $\rho(i)$ is primed, if the equality $e(C_0, D_{3,i}) = e(C_{\rho(i)}, D_{2,i})$ does not hold, it indicates that $t_{\rho(i)}$ is not equal to $w_{\rho(i)}$. Furthermore, we can have:

$$U_i = e(D_{1,i}, C_0)/e(D_{3,i}, C_0) = \frac{e(g^{A_i \cdot \upsilon}(u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i}, (gh)^s \cdot Z_0)}{e((u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{3,i}, (gh)^s \cdot Z_0)} = e(g,g)^{sA_i \cdot \upsilon}.$$

Thus, we can have:

$$\prod_{i \in I} (U_i)^{\sigma_i} = \prod_{i \in I} (e(g,g)^{sA_i \cdot \upsilon})^{\sigma_i} = e(g,g)^{\Sigma_{i \in I} sA_i \cdot \upsilon \sigma_i} = e(g,g)^{\alpha s} = \widetilde{C}.$$

4.4 Discussion

In Lai et al. [6], the searching keywords are disclosed in the trapdoor, which will cause a Test algorithm executor (i.e., an email gateway) to learn whether the encrypted data contains the keywords in the trapdoor. However, ESASE just discloses whether the keywords in the trapdoor are primed or not. We note that it might be inevitable in an ASE scheme that can support negation search operation, but it is acceptable if the scheme is proved secure in the security model defined in Section 3.2.

5 Security

Theorem 1. *If Assumptions 1, 2 and 3 hold, then ESASE is secure.*

Proof. The proof consists of the following three steps:

1. **Constructing the semi-functional ciphertexts and keys**.

   We define two additional structures: semi-functional ciphertexts and keys, which will not be used in the real system, but will be needed in our proof.

   Semi-functional Ciphertext. A semi-functional ciphertext is formed as follows. Firstly, a normal ciphertext $C'_W = (\widetilde{C}', C'_0, \{C'_i\}_{1 \le i \le n})$ is generated

by the Encrypt algorithm. Then, for each component $C_i^{'}$, a random value $\gamma_i \in \mathbb{Z}_N$ is chosen. Let g_2 be a generator of $\mathbb{G}_{p_2}$ and $c \in \mathbb{Z}_N$ be a random exponent, the semi-functional ciphertext $C_W = (\widetilde{C}, C_0, \{C_i\}_{1 \le i \le n})$ is set to be:

$$\widetilde{C} = \widetilde{C}^{'}, C_0 = C_0^{'} \cdot g_2^c, C_i = C_i^{'} \cdot g_2^{c\gamma_i}.$$

Note that the values $\{\gamma_i\}_{1 \le i \le n}$ are chosen randomly once and then fixed in the following semi-functional keys.

Semi-functional Key. To create a semi-functional key, a normal trapdoor key $T_{\mathcal{P}}^{'} = ((\mathbf{A}, \rho, \mathcal{F}), \{D_{1,i}^{'}, D_{2,i}^{'}, D_{3,i}^{'}\}_{1 \le i \le \ell})$ should be first generated by the Trapdoor algorithm. Then, it chooses a random vector $z \in \mathbb{Z}_N^m$ and sets $\delta_i = A_i \cdot z$. For each row i of the $\ell \times m$ matrix $\mathbf{A}$, a random value $\eta_i \in \mathbb{Z}_N$ is chosen. Finally, a semi-functional key $T_{\mathcal{P}} = ((\mathbf{A}, \rho, \mathcal{F}), \{D_{1,i}, D_{2,i}, D_{3,i}\}_{1 \le i \le \ell})$ is divided into two types of forms. The type 1 is set to be:

$$D_{1,i} = D_{1,i}^{'} \cdot g_2^{\delta_i + \eta_i \gamma_{\rho(i)}}, D_{2,i} = D_{2,i}^{'} \cdot g_2^{\eta_i}, D_{3,i} = D_{3,i}^{'} \cdot g_2^{\eta_i \gamma_{\rho(i)}}.$$

The type 2 is similarly formed as:

$$D_{1,i} = D_{1,i}^{'} \cdot g_2^{\delta_i}, D_{2,i} = D_{2,i}^{'}, D_{3,i} = D_{3,i}^{'}.$$

2. **Defining a sequence of attack games**.

We organize our proof as a sequence of games. The first game, $Game_{real}$, is the real security game defined in Section 3.2. In the next game, $Game_0$, all of the trapdoor keys are normal, but the challenge ciphertext is semi-functional. Let q be the number of trapdoor key queries made by the attacker. For k from 1 to q and ξ from 1 to n, we define:

- $Game_{k,1}$: the first $k-1$ trapdoor keys are semi-functional of type 2, the k^{th} trapdoor key is semi-functional of type 1, and the remaining trapdoor keys are normal. In addition, the challenge ciphertext is semi-functional.
- $Game_{k,2}$: the first k trapdoor keys are semi-functional of type 2, and the remaining trapdoor keys are normal. In addition, the challenge ciphertext is semi-functional.
- $Game_{final,\xi}$: all the trapdoor keys are semi-functional of type 2, and the challenge ciphertext $C_{W_\beta} = (\widetilde{C}, C_0, \{C_1, \ldots, C_n\})$ is a semi-functional encryption of W_β with $C_1, \ldots, C_\xi$, each of which is randomly chosen from $\mathbb{G}_{p_1 p_2 p_4}$.

Note that $Game_{0,2}$ and $Game_{final,0}$ can be considered as another way of denoting $Game_0$ and $Game_{q,2}$ respectively.

3. **Proving these attack games are indistinguishable with the real game**.

We prove that these games are indistinguishable in four lemmas, whose formal descriptions and proofs are given in the Appendix. It is clear that in

$Game_{final,n}$, the attacker's advantage is negligible since the challenge ciphertext C_{W_β} is independent of the keyword sets W_0 and W_1. Therefore, we conclude that the advantage of the adversary in $Game_{real}$ is negligible. This completes the proof of Theorem 1.

6 Efficiency

We compare our proposed scheme with Katz et al. [12] and Lai et al. [6] in Table 1.

Storage and Communication Overhead. The storage and communication overhead is mainly determined by the sizes of the ciphertext and trapdoor respectively. Table 1 shows the size of a ciphertext (resp. a trapdoor), both in ESASE and [6], is *linear* with the number of keyword fields n (resp. the number of rows ℓ) rather than *superpolynomial* in [12].

Computation Overhead. The computation overhead is mainly determined by three algorithms, including Encryption, Trapdoor, and Test. Table 1 shows that the computation overhead of Encryption (resp. Trapdoor), both in ESASE and [6], is *linear* with n (resp. ℓ), not *superpolynomial*. In addition, since ESASE supports the negation search, the Test cost of it is a little bigger than [6].

It can be concluded that the efficiency of ESASE is close to Lai et al. [6], but much better than Katz et al. [12]. Furthermore, compared with these two schemes, ESASE can simultaneously support conjunctive, disjunctive and negation search operations.

Table 1: Comparisons of efficiency of existing expressive ASE schemes.

Schemes	Katz et al. [12]	Lai et al. [6]	ESASE
Size of ciphertext	$(1+2m)L_1$	$(1+n)L_1 + L_2$	$(1+n)L_1 + L_2$
Size of trapdoor	$(1+2m)L_1$	$2\ell \cdot L_1$	$3\ell \cdot L_1$
Encryption	$(1+4m) \cdot \mathsf{e}$	$2(1+n) \cdot \mathsf{e}$	$(2+n) \cdot \mathsf{e}$+p
Trapdoor	$6m \cdot \mathsf{e}$	$4\ell \cdot \mathsf{e}$	$4\ell \cdot \mathsf{e}$
Test	$(1+2m) \cdot \mathsf{p}$	$\leq \psi_1 \cdot \mathsf{e} + 2\psi_2 \cdot \mathsf{p}$	$\leq \psi_1 \cdot \mathsf{e} + 2(\psi_2+\psi_3) \cdot \mathsf{p}$
Security	secure under the standard model	less secure	secure under the standard model
Expressiveness	AND, OR	AND, OR	AND, OR, NOT

[1] n: the number of keyword fields; ℓ: the number of rows in the matrix $\mathbf{A}$; L_1 (or L_2): a bit length of a group element in $\mathbb{G}$ (or $\mathbb{G}_T$); m: the length of the vector corresponding to the ciphertext in Katz et al. [12]. Note that m is proportional to d^ℓ, where d is the maximum degree (of the resulting polynomial) in each variable in [12].

[2] e: an exponentiation operation in $\mathbb{G}$ or $\mathbb{G}_T$; p: a pairing operation; ψ_1: the number of elements in $I_{\mathbf{A},\rho} = \{I_1, \ldots, I_{\psi_1}\}$; ψ_2: $|I_1| + \ldots + |I_{\psi_1}|$; ψ_3: the number of primed keywords in a search predicate.

7 Extensions

7.1 Range Search

Besides the boolean search, range search is also an important requirement for the searchable encryption. However, there is no expressive ASE scheme for the range search. Here we show how ESASE could be extended to support a class of simple range search [25]. Although supporting an arbitrary range search can be achieved by using the technique in [26], high complexity will be introduced when there are lots of disjunctive operations in a predicate. Therefore, we are not aiming at it.

We first introduce the concept of the keyword hierarchy. Let Z_i be a numerical keyword field, a keyword hierarchy over Z_i is a balanced tree $\Gamma(Z_i)$, where each internal node represents a range that is the union of the ranges of its children nodes and each leaf node is the keyword. In addition, we define some notations as follows.

- $\mathbb{R}(id)$: the range of a internal node id.
- $\mathbb{P}(z)$: the path from a leaf node z to the root. For every node id in $\mathbb{P}(z)$, it has $z \in \mathbb{R}(id)$.
- $\Gamma_l(Z_i)$: the node set in the l-th level of the tree, it is also called a "*level-l field*". Any node in $\Gamma_l(Z_i)$ is called a "*level-l keyword*" and denotes a "*level-l simple range*".

Suppose there is a "*Hour*" field Z_4 with keywords from number 0 to 23, Figure 4 depicts the keyword hierarchy. The path of the leaf node "12" is ("0-23", "12-17", "12-13", "12"), and "12-17", "12-13" are level-2 and level-3 simple range respectively.

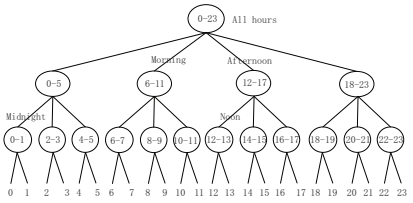

Fig. 4: The keyword hierarchy of the "*Hour*" field.

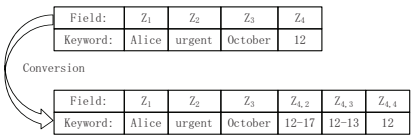

Fig. 5: An example of field structure conversion for the "*Hour*" field.

To support the simple range search based on the keyword hierarchy in E-SASE, the original field structure should be converted in the following way: for each hierarchical field Z_i, let φ be its maximum level and z_i be a leaf node, we expand Z_i into $(\varphi - 1)$ subfields: $Z_{i,2}$, $\cdots$, $Z_{i,\varphi}$, where the value set of $Z_{i,l}$ is $\Gamma_l(Z_i)$ and the value of $Z_{i,l}$ is the l-th element in $\mathbb{P}(z_i)$. Note that since there is only one keyword in $Z_{i,1}$, we ignore this subfield. Figure 5 shows that the

Hour field is expanded into 3 subfields. As a result, the parameter n in the Setup algorithm will be set to 6 and the keyword set to be encrypted will be {"*Alice*", "*urgent*", "*October*","12−17","12−13","12"}.

Moreover, before we use a predicate with range search to generate a trapdoor, we should convert it based on the system-defined simple ranges. For example, an original search predicate (Z_1 = "*Alice*" AND ("6" $\leq Z_4 \leq$ "13")) can be converted into (Z_1 = "*Alice*" AND ($Z_{4,2}$ = "6-11" OR $Z_{4,3}$ = "12-13")). Since ESASE supports the negation search, a search predicate (Z_1 = "*Alice*" OR (NOT "6" $\leq Z_4 \leq$ "13")) can be converted into (Z_1 = "*Alice*" OR ($Z_{4,2}$ = "(6-11)′" AND $Z_{4,3}$ = "(12-13)′")).

We note this class of simple range search will bring two benefits. On the one hand, the hierarchy on a numerical field can be well-designed depending on different usages. Take Figure 4 for example, the node "6-11" stands for the morning and "12-17" for the afternoon. On the other hand, a user can choose the search ranges based on different choices of granularity.

7.2 Multi-user Setting

We briefly outline how we extend ESASE scheme to the multi-user setting [9]. In a Single-user ESASE scheme, if a sender wants to send an encrypted email to multiple receivers in the example of a secure email system, he needs to encrypt the same email and the same keywords with each receiver's public key respectively. A multi-user ESASE scheme can eliminate these repeated operations and minimize the size of the public key material that needs to be obtained. We use the *randomness re-use* technique [27, 28] as follows.

We observe that in the Trapdoor algorithm, the parameter α is the secret in the LSSS representation $\mathbb{A} = (\mathbf{A}, \rho, \mathcal{E}, \mathcal{F})$. In addition, only one element, g^{α} in the public key depends upon α. Therefore, we can use different α to distinguish between different users. Specifically, in the Setup algorithm of a multi-user ESASE scheme, the public parameter PP shared by all the users is published as:

$$PP = N, g, \{H_i\}_{1 \leq i \leq n}, X_4.$$

while the public/secret key pair $\{pk_x, sk_x\}$ for user x is:

$$\{pk_x, sk_x\} = \{g^{\alpha_x}, (u_1, \ldots, u_n, \alpha_x, X_3)\}.$$

Note that all the users also share the same parameters $u_1, \ldots, u_n$, and X_3.

When a sender is to encrypt a keyword set W for k users, he first uses the public parameter PP to encrypt W to obtain the components C_0 and $\{C_i\}_{1 \leq i \leq n}$, and then uses each user's public key pk_x to generate the components $\widetilde{C}_1$, ..., $\widetilde{C}_x$, ..., $\widetilde{C}_k$ respectively. Finally, for the user x ($1 \leq x \leq k$), the corresponding ciphertext C_W will be:

$$C_W = (\widetilde{C}_x, C_0, \{C_i\}_{1 \leq i \leq n}).$$

The rest two algorithms of the multi-user ESASE scheme are the same with ESASE respectively. We compare the efficiency of the Single-user and Multi-user ESASE in Table 2. It shows that the multi-user ESASE is more efficient when there are multiple receivers.

Table 2: Comparisons of efficiency for the Single-user and Multi-user ESASE.

Schemes	Single-user ESASE	Multi-user ESASE
OE	$(2+n)\cdot k\mathsf{e}+k\cdot\mathsf{p}$	$(1+n+k)\cdot\mathsf{e}+k\cdot\mathsf{p}$
SPK	$(3+n)k\cdot L_1$	$(2+n+k)\cdot L_1$

k: the number of the receivers; n, e, p, L_1: defined as in Section 6; OE: the encryption overhead; SPK: the size of the public key material that a sender needs to obtain.

8 Conclusion and Future Work

Searchable encryption is an important cryptographic mechanism which enables to perform secure searches over encrypted data stored on untrusted servers. In this paper, we present a new scheme ESASE, to solve the problem of expressive search in the public key setting. To the best of our knowledge, ESASE is the first that can simultaneously support conjunctive, disjunctive and negation search operations. Then we prove ESASE is secure under the standard model. By analyzing, the efficiency of ESASE compares favorably to that of existing, less-expressive asymmetric searchable encryption schemes. Finally, we make two useful extensions: one is to support the range search, and another is to the multi-user setting. A further direction is to find more efficient schemes that can achieve the expressive search in the public key setting.

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searcheson encrypted data. In: IEEE Symposium on Security and Privacy, S&P 2000, pp. 44–55 (2000)
2. Goh, E.J.: Secure indexes. IACR Cryptology ePrint Archive 2003, 216 (2003)
3. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: ACM CCS 2006, pp. 79–88 (2006)
4. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
6. Lai, J., Zhou, X., Deng, R. H., Li, Y., Chen, K.: Expressive search on encrypted data. In: ACM ASIACCS 2013, pp. 243-252 (2013)

7. Moataz, T., Shikfa, A.: Boolean symmetric searchable encryption. In: ACM ASIACCS 2013, pp. 265-276 (2013)
8. Chen, Z., Wu, C., Wang, D., Li, S.: Conjunctive keywords searchable encryption with efficient pairing, constant ciphertext and short trapdoor. In: Chau et al. (eds.) PAISI 2012. LNCS, vol. 7299, pp. 176–189. Springer, Heidelberg (2012)
9. Hwang, Y. H., Lee, P. J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi et al. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007)
10. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
11. Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In: Network and Distributed System Security Symposium (NDSS). (2012)
12. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
13. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In ACM CCS 2012, pp. 965–976 (2012)
14. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Financial Cryptography and Data Security (FC). (2013)
15. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing et al. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)
16. Park, D. J., Kim, K., Lee, P. J.: Public key encryption with conjunctive field keyword search. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 73–86. Springer, Heidelberg (2004)
17. Chang, Y.C., Mitzenmacher, M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
18. Cash, D., Jarecki, S., Jutla, C. S., Krawczyk, H., Rosu, M., Steiner, M.: Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries. In Canetti, R., Garay, J. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013)
19. Islam, M., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: Network and Distributed System Security Symposium (NDSS). (2012)
20. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In ACM CCS 2007, pp. 195–203 (2007)
21. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: IEEE Symposium on Security and Privacy, S&P 2010, pp. 273–285 (2010)
22. Attrapadung, N., Libert, B.: Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 384–402. Springer, Heidelberg (2010)
23. Beimel, A.: Secure schemes for secret sharing and key distribution (Doctoral dissertation. PhD thesis, Israel Institute of Technology. (1996)
24. Boneh, D., Goh, E. J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)

25. Li, M., Yu, S., Cao, N., Lou, W.: Authorized private keyword search over encrypted data in cloud computing. In: IEEE Distributed Computing Systems (ICDCS), pp. 383–392 (2011).
26. Lewko A., Waters B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)
27. Kurosawa, K.: Multi-recepient public-key encryption with shortend ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
28. Bellare M., Boldyreva A., Staddon J.: Randomness re-use in multi-recipient encryption schemeas. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2003)

A Security Proofs

Lemma 1. *If there is an algorithm $\mathcal{A}$ that distinguish the $Game_{real}$ and $Game_0$ with non-negligible advantage ϵ. Then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ is given $\{g, X_3, X_4, T\}$ and will simulate $Game_{real}$ or $Game_0$ with $\mathcal{A}$.

Setup. $\mathcal{B}$ chooses $h_1, \ldots, h_n \in \mathbb{G}_{p_4}$ and $\alpha, a_1, \ldots, a_n \in \mathbb{Z}_N$ uniformly at random. Then, it sets $u_1 = g^{a_1}, \ldots, u_n = g^{a_n}$, $H_1 = u_1 h_1, \ldots, H_n = u_n h_n$, and sends $\mathcal{A}$ the public key $pk = \{N, g, g^{\alpha}, \{H_i\}_{1 \le i \le n}, X_4\}$.

Phase 1. $\mathcal{B}$ can respond to trapdoor key requests by running the Trapdoor algorithm to make normal trapdoor keys because it knows the secret key $sk = \{u_1, \ldots, u_n, \alpha, X_3\}$.

Challenge. $\mathcal{A}$ sends $\mathcal{B}$ two keyword sets W_0, W_1. $\mathcal{B}$ first chooses random values $h, \widetilde{Z}_0, \{\widetilde{Z}_{1,i}\}_{1 \le i \le n} \in \mathbb{G}_{p_4}$. Then, it randomly chooses a keyword set W_β from $\{W_0, W_1\}$. Let $W_\beta = \{w_{\beta,1}, \ldots, w_{\beta,n}\}$, it computes $\widetilde{C} = e(g^{\alpha}, T)$, $C_0 = T \cdot \widetilde{Z}_0$, $C_i = T^{a_i w_{\beta,i}} \cdot \widetilde{Z}_{1,i}$. $\mathcal{B}$ sets $C_{W_\beta} = (\widetilde{C}, C_0, \{C_i\}_{1 \le i \le n})$ and sends it to $\mathcal{A}$ as the challenge ciphertext.

Phase 2. Same as Phase 1 with the restriction that none of the query trapdoors can be satisfied by W_0 and W_1.

Guess. The adversary outputs a guess β' of β:

- If $T \in \mathbb{G}_{p_1 p_2}$, let $T = g^s g_2^c$, then

$$\widetilde{C} = e(g, g^{\alpha})^s, C_0 = (gh)^s Z_0 \cdot g_2^c, C_i = (H_i{}^{w_{\beta,i}})^s Z_{1,i} \cdot g_2^{c\gamma_i},$$

  where $Z_0 = \widetilde{Z}_0 \cdot h^{-s}$, $Z_{1,i} = \widetilde{Z}_{1,i} h_i^{-(w_{\beta,i} s)}$, $\gamma_i = a_i w_{\beta,i}$. This is a semi-functional ciphertext and $\mathcal{B}$ simulates $Game_0$. Since the values of a_i, $w_{\beta,i}$ modulo p_1 are uncorrelated from their values modulo p_2, it is properly distributed.
- if $T \in \mathbb{G}_{p_1}$, this is a normal ciphertext and $\mathcal{B}$ simulates $Game_{real}$.

Therefore, if $\mathcal{A}$ can distinguish the $Game_{real}$ and $Game_0$ with non-negligible advantage ϵ, $\mathcal{B}$ can break Assumption 1 with advantage ϵ.

Lemma 2. *If there is an algorithm $\mathcal{A}$ that distinguish the $Game_{k-1,2}$ and $Game_{k,1}$ with non-negligible advantage ϵ. Then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ is given $\{g, X_1X_2, Y_2Y_3, X_3, X_4, T\}$ and will simulate $Game_{k-1,2}$ or $Game_{k,1}$ with $\mathcal{A}$.

Setup. Same as in Lemma 1.

Phase 1. Note that $\mathcal{B}$ knows the secret key $sk = \{u_1, \ldots, u_n, \alpha, X_3\}$. Considering a predicate $(\mathbf{A}, \rho\ , \mathcal{E} = \{t_{\rho(1)}, \ldots\ , t_{\rho(\ell)}\}, \mathcal{F} = \{f_{\rho(1)}, \ldots\ , f_{\rho(\ell)}\})$, $\mathcal{B}$ creates the trapdoor keys as follows.

- For $j < k$, $\mathcal{B}$ chooses a random vector υ such that $\mathbf{1} \cdot \upsilon = \alpha$, a random vector z', random exponents $r_i \in \mathbb{Z}_N$, random elements $V_{1,i}, V_{2,i}, V_{3,i} \in \mathbb{G}_{p_3}$, and set:

  $$D_{1,i} = g^{A_i \cdot \upsilon}(u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i}(Y_2Y_3)^{A_i z'}, D_{2,i} = g^{r_i} \cdot V_{2,i}, D_{3,i} = (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{3,i}.$$

  We note that this is a properly distributed semi-functional key of type 2 because the value of $A_i z'$ modulo p_2 is uncorrelated to its value modulo p_3.
- For $j > k$, since $\mathcal{B}$ knows the secret key sk, he can create a normal trapdoor key by running the trapdoor generation algorithm.
- For the k^{th} trapdoor key, $\mathcal{B}$ chooses a random vector υ' such that $\mathbf{1} \cdot \upsilon' = \alpha$, a random vector z such that $z \cdot \mathbf{1} = 0$, random exponents $\widetilde{\eta}_i \in \mathbb{Z}_N$, random elements $\widetilde{V}_{1,i}, \widetilde{V}_{2,i}, \widetilde{V}_{3,i} \in \mathbb{G}_{p_3}$, and sets:

  $$D_{1,i} = g^{A_i \cdot \upsilon'} T^{A_i z + \widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{1,i}, D_{2,i} = T^{\widetilde{\eta}_i} \widetilde{V}_{2,i}, D_{3,i} = T^{\widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{3,i}.$$

  Thus, if $T \in \mathbb{G}_{p_1p_2p_3}$, then T can be written as $g^r g_2^d V$, and

  $$\begin{aligned} D_{1,i} &= g^{A_i \cdot \upsilon}(u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i} \cdot g_2^{\delta_i + \eta_i \gamma_{\rho(i)}}, \\ D_{2,i} &= g^{r_i} \cdot V_{2,i} \cdot g_2^{\eta_i}, \\ D_{3,i} &= (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{3,i} \cdot g_2^{\eta_i \gamma_{\rho(i)}}, \end{aligned}$$

  where $\upsilon = \upsilon' + zr$, $r_i = r\widetilde{\eta}_i$, $\delta_i = dA_i z$, $\gamma_{\rho(i)} = a_i t_{\rho(i)}$, $\eta_i = d\widetilde{\eta}_i$, $V_{1,i} = V^{A_i z + \widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{1,i}$, $V_{2,i} = V^{\widetilde{\eta}_i} \widetilde{V}_{2,i}$, $V_{3,i} = V^{\widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{3,i}$. This is a semi-function key of type 1. Since the values of $\widetilde{\eta}_i, a_i, t_{\rho(i)}$ modulo p_1 are are uncorrelated from their values modulo p_2, it is properly distributed. If $T \in \mathbb{G}_{p_1p_3}$, it is a properly distributed normal trapdoor key.

Challenge. $\mathcal{A}$ sends $\mathcal{B}$ two keyword sets W_0, W_1. $\mathcal{B}$ first chooses random values $h, \widetilde{Z}_0, \{\widetilde{Z}_{1,i}\}_{1 \le i \le n} \in \mathbb{G}_{p_4}$. Then, it randomly chooses a keyword set W_β from $\{W_0, W_1\}$. Let $W_\beta = \{w_{\beta,1}, \ldots, w_{\beta,n}\}$, it computes $\widetilde{C} = e(g^\alpha, X_1X_2)$, $C_0 =$

$(X_1X_2) \cdot \widetilde{Z}_0$, $C_i = (X_1X_2)^{a_i w_{\beta,i}} \cdot \widetilde{Z}_{1,i}$. $\mathcal{B}$ sets $C_{W_\beta} = (\widetilde{C}, C_0, \{C_i\}_{1 \le i \le n})$ and sends it to $\mathcal{A}$ as the challenge ciphertext. If we let $X_1X_2 = g^s g_2^c$, then

$$\widetilde{C} = e(g, g^\alpha)^s, C_0 = (gh)^s Z_0 \cdot g_2^c, C_i = (H_i{}^{w_{\beta,i}})^s Z_{1,i} \cdot g_2^{c\gamma_i},$$

where $Z_0 = \widetilde{Z}_0 \cdot h^{-s}$, $Z_{1,i} = \widetilde{Z}_{1,i} h_i^{-(w_{\beta,i} s)}$, $\gamma_i = a_i w_{\beta,i}$. This is a semi-functional ciphertext. Since the values of a_i, $w_{\beta,i}$ modulo p_1 are uncorrelated from their values modulo p_2, it is properly distributed.

Phase 2. Same as Phase 1 with the restriction that none of the query trapdoors can be satisfied by W_0 and W_1:

- If $T \in \mathbb{G}_{p_1p_2p_3}$, the k^{th} trapdoor key is properly distributed semi-functional key of type 1. Thus, $\mathcal{B}$ has properly simulated $Game_{k,1}$.
- if $T \in \mathbb{G}_{p_1p_3}$, the k^{th} trapdoor key is properly distributed normal key. Thus, $\mathcal{B}$ has properly simulated $Game_{k-1,2}$.

Therefore, if $\mathcal{A}$ can distinguish the $Game_{k,1}$ and $Game_{k-1,2}$ with non-negligible advantage ε, $\mathcal{B}$ can break Assumption 2 with advantage ε.

Lemma 3. *If there is an algorithm $\mathcal{A}$ that distinguish the $Game_{k,1}$ and $Game_{k,2}$ with non-negligible advantage ϵ. Then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ is given $\{g, X_1X_2, Y_2Y_3, X_3, X_4, T\}$ and will simulate $Game_{k,1}$ or $Game_{k,2}$ with $\mathcal{A}$.

Setup. Same as in Lemma 1.

Phase 1. Note that $\mathcal{B}$ knows the secret key $sk = \{u_1, \ldots, u_n, \alpha, X_3\}$. Considering a predicate ($\mathbf{A}$, ρ , $\mathcal{E} = \{t_{\rho(1)}, \ldots, t_{\rho(\ell)}\}$, $\mathcal{F} = \{f_{\rho(1)}, \ldots, f_{\rho(\ell)}\}$), $\mathcal{B}$ creates the trapdoor keys as follows.

- For $j < k$ and $j > k$, the responses to trapdoor key queries are the same as in Lemma2.
- For the k^{th} trapdoor key, $\mathcal{B}$ chooses a random vector υ such that $\mathbf{1} \cdot \upsilon = \alpha$, a random vector z such that $z \cdot \mathbf{1} = 0$, random exponents $\widetilde{\eta}_i \in \mathbb{Z}_N$, random elements $\widetilde{V}_{1,i}$, $\widetilde{V}_{2,i}$, $\widetilde{V}_{3,i} \in \mathbb{G}_{p_3}$, and sets:

  $$D_{1,i} = g^{A_i \cdot \upsilon} (Y_2Y_3)^{A_i z} T^{\widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{1,i}, D_{2,i} = T^{\widetilde{\eta}_i} \widetilde{V}_{2,i}, D_{3,i} = T^{\widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{3,i},$$

  Thus, if $T \in \mathbb{G}_{p_1p_2p_3}$, then T can be written as $g^r g_2^d V$, and

  $$\begin{aligned} D_{1,i} &= g^{A_i \cdot \upsilon} (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i} \cdot g_2^{\delta_i + \eta_i \gamma_{\rho(i)}}, \\ D_{2,i} &= g^{r_i} \cdot V_{2,i} \cdot g_2^{\eta_i}, \\ D_{3,i} &= (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} \cdot V_{3,i} \cdot g_2^{\eta_i \gamma_{\rho(i)}}, \end{aligned}$$

  where $r_i = r\widetilde{\eta}_i$, $\delta_i = \log_{g_2} Y_2 \cdot A_i z$, $\gamma_{\rho(i)} = a_i t_{\rho(i)}$, $\eta_i = d\widetilde{\eta}_i$, $V_{1,i} = Y_3^{A_i z} V^{\widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{1,i}$, $V_{2,i} = V^{\widetilde{\eta}_i} \widetilde{V}_{2,i}$, $V_{3,i} = V^{\widetilde{\eta}_i a_i t_{\rho(i)}} \widetilde{V}_{3,i}$. This is a semi-function key of type 1.

Since the values of $\widetilde{\eta}_i, a_i, t_{\rho(i)}$ modulo p_1 are are uncorrelated from their values modulo p_2, it is properly distributed. If $T \in \mathbb{G}_{p_1 p_3}$, it is a properly distributed semi-functional key of type 2.

Challenge. Same as in LEMMA2.

Phase 2. Same as Phase 1 with the restriction that none of the query trapdoors can be satisfied by W_0 and W_1:

- If $T \in \mathbb{G}_{p_1 p_2 p_3}$, the k^{th} trapdoor key is properly distributed semi-functional key of type 1. Thus, $\mathcal{B}$ has properly simulated $Game_{k,1}$.
- if $T \in \mathbb{G}_{p_1 p_3}$, the k^{th} trapdoor key is is properly distributed semi-functional key of type 2. Thus, $\mathcal{B}$ has properly simulated $Game_{k,2}$.

Therefore, if $\mathcal{A}$ can distinguish the $Game_{k,1}$ and $Game_{k,2}$ with non-negligible advantage ε, $\mathcal{B}$ can break Assumption 2 with advantage ε.

Lemma 4. *If there is an algorithm $\mathcal{A}$ that distinguish the $Game_{final,\xi-1}$ and $Game_{final,\xi}$ with non-negligible advantage ϵ. Then we can construct an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Proof. $\mathcal{B}$ is given $\{g, g_2, X_2, uh', g^s B_{24}, X_3, X_4, T\}$ and will simulate $Game_{final,\xi-1}$ or $Game_{final,\xi-2}$ with $\mathcal{A}$.

Setup. $\mathcal{B}$ chooses $h_1, \ldots, h_{max\{1,\xi-1\}}, h_{\xi+1}, \ldots, h_n \in \mathbb{G}_{p_4}$ and $\alpha, a_1, \ldots, a_{max\{1,\xi-1\}}, a_{\xi+1}, \ldots, a_n \in \mathbb{Z}_N$ uniformly at random. Then, it sets $u_1 = g^{a_1}$, $\ldots$, $u_{max\{1,\xi-1\}} = g^{a_{max\{1,\xi-1\}}}$, $u_{\xi+1} = g^{a_{\xi+1}}$, $\ldots$, $u_n = g^{a_n}$, $H_1 = u_1 h_1$, $\ldots$, $H_{max\{1,\xi-1\}} = u_{max\{1,\xi-1\}} h_{max\{1,\xi-1\}}$, $H_\xi = uh'$, $H_{\xi+1} = u_{\xi+1} h_{\xi+1}$, $\ldots$, $H_n = u_n h_n$, and sends $\mathcal{A}$ the public key :

$$pk = \{N, g, g^\alpha, \{H_i\}_{1 \le i \le n}, X_4\}$$

Phase 1. Note that $\mathcal{B}$ knows the secret key $sk = \{u_1, \ldots, u_n, \alpha, X_3\}$. Consider $\mathcal{A}$ queries for a predicate $(\mathbf{A}, \rho, \mathcal{E} = \{t_{\rho(1)}, \ldots, t_{\rho(\ell)}\}, \mathcal{F} = \{f_{\rho(1)}, \ldots, f_{\rho(\ell)}\})$, $\mathcal{B}$ first chooses a random vector υ such that $\mathbf{1} \cdot \upsilon = \alpha$, a random vector z, random exponents $r_i \in \mathbb{Z}_N$, random elements $\widetilde{V}_{1,i}, \widetilde{V}_{2,i}, \widetilde{V}_{3,i} \in \mathbb{G}_{p_3}$, and then set: $D_{1,i} = g^{A_i \cdot \upsilon} (uX_2)^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i} \cdot g_2^{A_i z}$, if $\rho(i) = \xi$, or $D_{1,i} = g^{A_i \cdot \upsilon} (g^{a_{\rho(i)}})^{t_{\rho(i)} \cdot r_i} \cdot V_{1,i} \cdot g_2^{A_i z}$ otherwise. $D_{2,i} = g^{r_i} \cdot V_{2,i}$, $D_{3,i} = (g^{a_{\rho(i)}})^{t_{\rho(i)} \cdot r_i} \cdot V_{3,i}$. Thus, $D_{1,i}$ can be written as $g^{A_i \cdot \upsilon} (u_{\rho(i)})^{t_{\rho(i)} \cdot r_i} V_{1,i} \cdot g_2^{\delta_i}$, where $\delta_i = A_i z + t_{\rho(i)} r_i \log_{g_2} X_2$ if $\rho(i) = \xi$ or $\delta_i = A_i z$ otherwise. Note this is a properly distributed semi-functional key of type 2.

Challenge. $\mathcal{A}$ sends $\mathcal{B}$ two keyword sets W_0, W_1. $\mathcal{B}$ first chooses random values $h, \widetilde{Z}_0, \{\widetilde{Z}_{1,i}\}_{\xi \le i \le n} \in \mathbb{G}_{p_4}$. Then, it randomly chooses a keyword set W_β from $\{W_0, W_1\}$ and random elements $C_1, \ldots, C_{max\{1,\xi-1\}} \in \mathbb{G}_{p_1 p_2 p_4}$. Let $W_\beta = \{w_{\beta,1}, \ldots, w_{\beta,n}\}$, it computes $\widetilde{C} = e(g, g^s B_{24})$, $C_0 = g^s B_{24} \cdot \widetilde{Z}_0$, $C_\xi = (g^s B_{24})^{w_{\beta,\xi}} \cdot T \cdot \widetilde{Z}_{1,\xi}$, $\{C_i = g^s B_{24}{}^{a_i w_{\beta,i}} \cdot \widetilde{Z}_{1,i}\}_{\xi \le i \le n}$. $\mathcal{B}$ sets $C_{W_\beta} = (\widetilde{C}, C_0, \{C_i\}_{1 \le i \le n})$ and sends it to $\mathcal{A}$ as the challenge ciphertext.

Phase 2. Same as Phase 1 with the restriction that none of the query trapdoors can be satisfied by W_0 and W_1.

Guess. The adversary outputs a guess $\beta^{'}$ of β:

- Denote B_2, B_4 to be the $\mathbb{G}_{p_2}$, $\mathbb{G}_{p_4}$ parts of B_{24} respectively, D_2, D_4 to be the $\mathbb{G}_{p_2}$, $\mathbb{G}_{p_4}$ parts of D_{24} respectively. If $T = u^s D_{24}$, then

$$\widetilde{C} = e(g, g^{\alpha})^s, C_0 = (gh)^s Z_0 \cdot g_2^c, \{C_i = (H_i{}^{w_i})^s Z_{1,i} \cdot g_2^{c\gamma_i}\}_{\xi \le i \le n},$$

  where $c = \log_{g_2} B_2$, $\gamma_\xi = (\log_{g_2}(B_2^{w_{\beta,\xi}} D_2))/c$, $\{\gamma_i = a_i w_{\beta,i}\}_{\xi \le i \le n}$, $Z_0 = B_4 \widetilde{Z}_0 \cdot h^{-s}$, $Z_{1,\xi} = B_4^{w_{\beta,i}} D_4 \widetilde{Z}_{1,\xi} (uh')^{-(w_{\beta,i} s)}$, $\{Z_{1,i} = B_4^{a_i w_{\beta,i}} \widetilde{Z}_{1,i} h_i^{-(w_{\beta,i} s)}\}_{\xi \le i \le n}$. Since the values of a_i, $w_{\beta,i}$ modulo p_1 are uncorrelated from their values modulo p_2, it is properly distributed semi-functional ciphertext with $C_1, \ldots, C_{max\{1,\xi-1\}}$ random in $\mathbb{G}_{p_1 p_2 p_4}$. Thus, $\mathcal{B}$ has properly simulated $Game_{final,\xi-1}$.
- if $T \in \mathbb{G}_{p_1 p_2 p_4}$, this is a properly distributed semi-functional ciphertext with $C_1, \ldots, C_\xi$ random chosen from $\mathbb{G}_{p_1 p_2 p_4}$. Thus, $\mathcal{B}$ has properly simulated $Game_{final,\xi}$.

Thus, if $\mathcal{A}$ can distinguish the $Game_{fina,\xi-1}$ and $Game_{final,\xi}$ with non-negligible advantage ε, $\mathcal{B}$ can break Assumption 3 with advantage negligibly ε.