Keyword Search over Shared Cloud Data without Secure Channel or Authority

Yilun Wu[†‡], Jinshu Su[†], and Baochun Li[‡]
[†] College of Computer, National University of Defense Technology, Changsha, Hunan, China
[‡] Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada
yilun@ece.utoronto.ca, sjs@nudt.edu.cn, bli@ece.toronto.edu

Abstract—**Storage services play an important role in a public cloud. By outsourcing data to the remote cloud, users do not need to maintain a local storage infrastructure and can significantly lower the storage cost. To protect the privacy, documents must be encrypted before outsourcing. This raises a new challenge for the document owner: how should the encrypted documents be securely searched in a public cloud? While many mechanisms have been proposed to support secure search over the encrypted documents, most of these mechanisms require secure channels to transmit the secret information, such as the secret keys and trapdoors, and is difficult to deploy in cloud systems. Moreover, some existing mechanisms require an authority to control the access requests of users, which inevitably increases the complexity of cloud infrastructure.**

This paper considers a more stringent security model where an eavesdropper exists in the cloud and can eavesdrop on all transmission channels. We propose a novel mechanism that supports multi-user keyword search over the encrypted data without relying on any secure channel or authority. The eavesdropper can neither forge valid trapdoors from the intercepted information nor can it directly use the intercepted trapdoors to complete the keyword search. Security analysis shows that the proposed mechanism is secure.

Index Terms—**keyword search; cloud security; secure channel;**

I. Introduction

The past few years have witnessed an explosive growth of cloud storage services. Thanks to the economies of scale of datacenter, cloud storage service provides a more convenient and cost efficient solution for a data owner than the traditional solution of maintaining an in-house storage infrastructure. Despite all these benefits, outsourcing data to a remote public cloud puts the data privacy at risk. To protect the privacy, data has to be encrypted before outsourcing [1]. However, this raises a new challenge of data searching and retrieving.

To answer this challenge, several mechanisms, referred to as *searchable encryption*, have been proposed to enable users to search over the encrypted data to retrieve the requested documents [2], [3], [4]. These mechanisms allow only the data owner to perform document search. However, more often than not, a data owner might want to share its data with multiple *data users* who should also be given the permission of searching over the encrypted data. This requirement is referred to as the *keyword search with multi-user setting*. To meet this requirement, existing mechanisms, such as [5], [6], [7], require data users to obtain a certain type of secret information (e.g., trapdoors or secret keys used to generate trapdoors) from the data owner to generate some valid trapdoors for certain keywords. However, there are several drawbacks of these mechanisms, which we elaborate below.

First, to protect the secret information from being revealed, some secure transmission channels must be established in the existing mechanisms. This may lead to a potential safety hazard. If an adversary can eavesdrop on these channels, it would obtain the secret information, with which it can search over the encrypted data without the permission of the data owner, cracking the entire system. Worse, existing multi-user mechanisms cannot defense against such an eavesdropper. Moreover, the number of secure channels needed depends on the number of data users. Establishing such a large number of secure channels is very expensive for a data owner [8]. This is true for key management as well, given that the number of transmitted keys also depends on the number of data users. In addition, some existing mechanisms (e.g., [9], [10]) require an authentication authority to guard against unauthorized access to the encrypted documents. While such an authority helps the access control, it introduces a significant complexity to the cloud infrastructure.

In light of the problems above, we consider a more stringent security scenario where all transmission channels could be potentially bugged. Specifically, we allow an eavesdropper who can have access to all the transmitted information by eavesdropping on all the transmission channels in a cloud. In the presence of such an eavesdropper, no existing mechanism is secure any more because data users cannot securely obtain trapdoors from the data owner. To solve this problem, we propose a novel mechanism for keyword search without relying on any secure channel or authentication authority. Our mechanism utilizes a recently proposed cryptographic primitive called *Non-Interactive Key Exchange based on indistinguishability obfuscation* (i.e., $i\mathcal{O}$-based NIKE) [11] to share secret information via insecure transmission channels. We also design a new search protocol without an authentication authority. Our contributions are summarized as follows.

- Unlike existing works, the security guarantee of the proposed multi-user mechanism for keyword search does not rely on any secure channel. No secret information is revealed to an eavesdropper.
- We design a new search protocol to deny unauthorized requests of keyword search. The new protocol does not require any authentication authority for access control.

This significantly reduces the complexity of the cloud infrastructure.

- Our mechanism offers a stronger security guarantee than that of the existing works. Security analysis shows that our mechanism can guard against an eavesdropper who intends to search over the encrypted documents without the permission of the data owner. No matter what kind of information an eavesdropper collects by eavesdropping the transmission channels, its search request can never complete successfully.

The remainder of the paper is organized as follows. In Section II, we survey related works. In Section III, we present the system model, the security model, and the design objectives. The notations and cryptographic primitives are introduced in Section IV. Section V gives the details of the proposed mechanism, followed by the security analysis in Section VI. Section VII concludes the paper.

II. Related Work

The seminal work by Song et al. [2] is credited with the first practical searchable encryption mechanism. Since then, many follow-up mechanisms have been proposed in the literature [3], [4], [12], [13], [14], [15], [16]. All these mechanisms allow only the data owner to search over the encrypted documents, and hence cannot be applied to data sharing services in a public cloud where the encrypted documents should also be searchable for authorized users. In light of this problem, many mechanisms have been proposed to support *multi-user* searchable encryption, with which data searching is not limited to the data owner, but is also enabled to authorized users [5], [7], [9], [10], [17], [18], [19], [20], [21]. For example, Wang et al. [6], [19], [21] proposed the ranked keyword search that ranks the searching results based on the keyword frequency. The mechanisms proposed in Li et al. [17] and Wang et al. [18] support fuzzy keyword search and return the closest possible documents if the searching keywords do not exactly match those predefined ones. The mechanisms proposed in [20], [21] combine the techniques of both fuzzy and ranked search and can search multiple keywords in one request. Recently, the attribute-based encryption has been used to design the fine-grained access control mechanisms [9], [10]. All the aforementioned multi-user keyword search mechanisms require a secure channel between the data owner and each data user to transmit secret information. In addition, many mechanisms, such as [7], [9], [10], manage access control via an authentication authority, adding more complexity to the existing cloud infrastructure. Unlike existing works, our mechanism does not rely on secure channels, and has no authentication authority for access control.

III. Problem Statement

In this section, we formulate the mechanism design problem. We shall begin with the system model by presenting the network architecture, entities, and possible interactions among these entities. We shall then present the security model with a strong adversary who can eavesdrop on all transmission channels in the cloud. We shall also clarify the design objectives of our mechanism with four desired security properties.

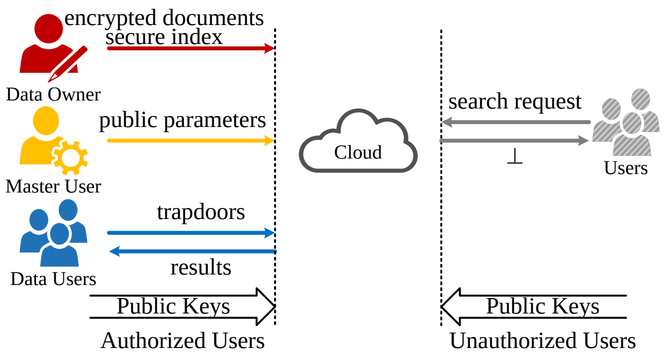

Fig. 1. System Model

A. System Model

Fig. 1 gives an overview of the system model. There are five entities in the system: the data owner, the master user, the data user, the unauthorized user, and the cloud server. Let S be the set of authorized users including the data owner, the master user, and the data users. Each cloud user, no matter in S or not, generates a key pair consisting of both a public key and a secret key. It then sends the public key to the cloud server, who stores it in a table that is publicly accessible to all cloud users. The master user generates some public parameters for the authorized users and outsources them to the cloud server. Each cloud user in the system can download these parameters from the cloud server, but only the authorized users can utilize these parameters to generate the valid shared symmetric keys. Now for the data owner, let D be the collection of documents it wants to share with the authorized users. The owner encrypts documents D based on the shared symmetric keys. The encrypted documents are denoted by E. The data owner also builds a secure index, denoted as I, containing all the keywords in the original documents D. Both the encrypted documents E and the secure index I are uploaded to the cloud server. To search over the encrypted documents, each authorized user downloads the public parameters from the cloud server. Together with its secret key, the user generates the shared symmetric keys, with which it can generate a valid trapdoor for a certain keyword. It can then search that keyword over the encrypted documents using the corresponding trapdoor. Note that the entire process does not work for an unauthorized user. While it can have access to the public parameters and the public keys, it cannot successfully compute the shared symmetric keys, and cannot generate a valid trapdoor for keyword search.

B. Security Model

Unlike existing works, we consider a new model of an adversary that plays two different roles at the same time: an unauthorized cloud user as well as an eavesdropper who can eavesdrop on the transmission channels between any two entities in the cloud. As a cloud user, the adversary has access to both the public keys of other users and the public parameters posted by the master user. As an eavesdropper, the adversary can sniff out information sent through all the transmission channels. In other words, there is no secure channel in the considered cloud environment. The purpose of attacking of the adversary is to successfully search over the encrypted documents without the permission of the data owner. In this paper, we consider two attack patterns from the adversary. The first is to intercept those valid trapdoors submitted by authorized users in S, with which the adversary tries to search over the encrypted documents using the intercepted trapdoors. The second attack pattern of the adversary is to forge a trapdoor based on the eavesdropped information. The adversary then sends the forged trapdoor to the cloud server for keyword search.

Following the existing works, the cloud server is assumed to be *honest-but-curious*. In particular, the cloud server honestly follows the designated protocol. At the same time, it tries to learn as much additional information about the keyword search by deducing the received information curiously.

It is worth emphasizing that our security model does not assume the attack of collusion between the eavesdropper and an authorized data user. Otherwise, the eavesdropper can directly obtain the shared symmetric keys from that colluded user. In our model, each cloud user keeps its own secret keys in private.

C. Design Objectives

Despite the presence of the strong adversary, we require the following security properties achieved in the designed mechanism.

1) Secure Key Sharing.: Despite the insecure transmission channels, the shared symmetric keys cannot be revealed to the eavesdropper.

2) No Authority.: The mechanism should not rely on any authentication authority to perform access control. Without such an authority, all cloud users can search over the encrypted documents. However, only those authorized users who hold the shared symmetric keys can get the correct searching results.

3) Indistinguishable Trapdoors.: The eavesdropper cannot distinguish two trapdoors of the same keyword even if the two were generated by the same user. This makes it difficult for the eavesdropper to build the access pattern.

4) Secure Search.: Only the authorized users can search over the shared and encrypted documents.

IV. Notations and Preliminaries

A. Notations

We introduce the following notations.

- U – the set of all cloud users. The number of users in U is denoted as num_U.
- S – the subset of U that contains the authorized users including the data owner, the master user, and the data users. The number of the users in S is denoted as num_S.
- K_{SI}, K_{SD} – the symmetric keys held by the users in S, where K_{SI} is used to generate the secure index, and K_{SD} is used to encrypt the documents.
- K_{Cu} – the key held between the cloud server and user u.
- K_{CO} – the key held between the cloud server and the data owner.
- T – the table that contains the public keys of the users. The size of T is denoted as num_T.
- D – the collection of documents that the data owner wants to share with authorized users, where the i^{th} document is denoted as D_i. In total, there are num_D documents to share with.
- E – the encrypted documents D.
- KW – the set of keywords in D. The size of KW is denoted as num_{KW}.
- $D(kw)$ – the subset of D that contains keyword kw. The encrypted $D(kw)$ is denoted as $E(kw)$.
- id(D_i) – the identifier of document D_i.
- td – the trapdoor.

B. Preliminaries

The following cryptographic techniques serve as the building blocks of our mechanism.

Indistinguishability Obfuscation: According to [22], we say a uniform probabilistic polynomial time (PPT) algorithm for a circuit class $\{\mathcal{C}_\lambda\}$ is an *indistinguishability obfuscator* ($i\mathcal{O}$) if the following two conditions hold:

- For any $C \in \mathcal{C}_\lambda$, let $C' = i\mathcal{O}(\lambda, C)$. We have Pr[$C'(x) = C(x)$]=1 for any input x.
- For any pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, if $C_0(x) = C_1(x)$ for all input x, then $i\mathcal{O}(\lambda, C_0)$ and $i\mathcal{O}(\lambda, C_1)$ are indistinguishable. According to the first condition, we see that $i\mathcal{O}(\lambda, C)$ and $i\mathcal{O}(\lambda, C')$ are indistinguishable.

In this paper, we use $i\mathcal{O}$-based NIKE protocol to share the symmetric keys with authorized users. The $i\mathcal{O}$-based NIKE protocol has many advantages [11] over other NIKE protocols, e.g., no trusted setup and short public values. Since the protocol does not require trusted setup, no secret key is exposed [11]. In particular, the $i\mathcal{O}$-based NIKE allows each authorized user to generate the shared symmetric keys based on its own secret key and some public parameters. No secret information is transmitted on channels.

Symmetric Encryption: The *Symmetric Key Encryption* (SKE) consists of three polynomial-time algorithms, i.e., $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Specifically, $\mathsf{SKE.Gen}$ takes a security parameter λ as input, and outputs a symmetric key K; $\mathsf{SKE.Enc}$ takes a key K and a plaintext m as inputs, and outputs a ciphertext c; $\mathsf{SKE.Dec}$ uses key K to decrypt c and outputs plaintext m. In our mechanism, SKE is used to generate the secure index and to encrypt documents. The functionality of $\mathsf{SKE.Gen}$ is replaced by the $i\mathcal{O}$-based NIKE protocol.

Digital Signature Scheme: The basic digital signature scheme consists of three algorithms, i.e., $\mathsf{DS} = \{\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}\}$. In particular, $\mathsf{DS.Gen}$ generates a key pair

(sk, pk); $\mathsf{DS.Sign}$ takes the secret key sk and a message m as inputs, and returns a signature σ; $\mathsf{DS.Ver}$ takes σ, m, and public key pk as inputs and returns *TRUE* if σ is calculated by the corresponding sk, otherwise, it returns *FALSE*. As shown in [11], the digital signature scheme is used to verify user identities.

Pseudorandom Function: The *pseudorandom function* (PRF) is a function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{K} \in \{0,1\}^*$ is the key space, $\mathcal{X} \in \{0,1\}^*$ the domain space, and $\mathcal{Y} \in \{0,1\}^*$ the range. Given $(k, x) \in \mathcal{K} \times \mathcal{X}$, a deterministic polynomial algorithm can efficiently compute $F(k, x) \in \mathcal{Y}$. A pseudorandom function is *secure* if for a randomly chosen key, no PPT adversary can distinguish the returned value of the pseudorandom function from the returned value of a random function with non-negligible probability.

Bilinear Map: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of a large prime p. Let g be a generator of $\mathbb{G}_1$. The map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map if the following conditions hold. 1) *Bilinearity*: for any $u, v \in \mathbb{G}_1$ and any $a, b \in \mathbb{Z}_p$, map $\hat{e}$ is bilinear if $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$. 2) *Non-degeneracy*: $\hat{e}(g, g) \neq 1$. If g is a generator of $\mathbb{G}_1$, then $\hat{e}(g, g)$ is a generator of $\mathbb{G}_2$. 3) *Computability*: for any $u, v \in \mathbb{G}_1$, there is an efficient algorithm to compute $\hat{e}(u, v)$.

The following cryptographic primitives shall be used in the security analysis.

Constrained Pseudorandom Function: As defined in [23], a constrained PRF can derive a constrained key k' corresponding to a subset $\mathcal{X}_1 \subseteq \mathcal{X}$. The constrained PRF can only evaluate the pseudorandom function when the input is (k', x), where $x \in \mathcal{X}_1$. A secure constrained PRF [11] is defined as follows. An adversary $\mathcal{A}$ is allowed to request a constrained pseudorandom function. However, $\mathcal{A}$ cannot distinguish the returned value of PRF from random with non-negligible probability if the chosen input is not in $\mathcal{X}_1$. In our mechanism, we assume that the constrained PRF is secure.

Constrained Signature: The definition of constrained signature follows [11]. Given a circuit C, a constrained signature computes a constrained public key pk_C by a constrained key generation algorithm $\mathsf{ConstrainGen}$. One can only use x such that $C(x) = 1$ to generate a valid signature $\sigma = \mathsf{Sign}(sk_C, x)$, and to pass verification $\mathsf{Ver}(pk_C, x, \sigma)$. We say a constrained signature is secure if the following condition holds. For any PPT adversary $\mathcal{A}$ that can query on all x such that $C(x) = 1$, the probability of distinguishing pk_C from a normal public key generated by $\mathsf{DS.Gen}$ is negligible. The constrained signature is assumed to be secure in our mechanism.

Computational Diffie-Hellman (CDH) Assumption: Let $\mathbb{G}$ be a cyclic group of a large prime p with a generator g. Given (p, g, g^a, g^b), where $a, b \in \mathbb{Z}_p$ are randomly chosen, the probability of computing g^{ab} is negligible for a PPT adversary.

Discrete Logarithm Assumption: Let $\mathbb{G}$ be a cyclic group of a large prime p with a generator g. Given (p, g, g^a), where a is randomly chosen, the probability of computing a is negligible for a PPT adversary.

V. The Proposed Mechanism

In this section, we present the details of the proposed mechanism. The $i\mathcal{O}$-based NIKE protocol is used to generate two shared symmetric keys for the authorized users. Once these keys have been generated, the data owner can generate a secure index I for the shared documents D and encrypt D as well. With these keys, authorized users can generate valid trapdoors for keywords.

A. Algorithm Definition

In this subsection, we describe seven algorithms which will be used to construct the mechanism.

- $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, p) \leftarrow \mathsf{Init}(1^\lambda)$: This algorithm takes a security parameter λ as input and returns $\mathbb{G}_1$, $\mathbb{G}_2$, p, g and $\hat{e}$. $\mathbb{G}_1$, $\mathbb{G}_2$ are two cyclic groups of a large prime p. g is a generator of $\mathbb{G}_1$. $\hat{e}$ is a bilinear map where $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
- $str_S \leftarrow \mathsf{Interpret}(S, U)$: The interpret algorithm takes S and U as inputs and returns a string str_S. str_S is the identifier of S.
- $(pk, sk) \leftarrow \mathsf{UKeyGen}(\lambda)$: The user key generation algorithm takes a secret parameter λ as input and invokes $\mathsf{DS.Gen}(\lambda)$ to return a key pair (pk, sk).
- K or $\perp \leftarrow \mathsf{SKeyGen}(MK_s, s_u)$: The symmetric key generation algorithm takes public parameters MK_s and a secret key s_u as inputs and returns a symmetric key K or $\perp$.
- $I \leftarrow \mathsf{IndexEnc}(D, K_{SI}, K_{SO})$: The index encryption algorithm takes D and two symmetric keys K_{SI},K_{SO} as inputs and returns an secure index I. The functionality of two keys will be discussed in next subsection.
- $td \leftarrow \mathsf{Trapdoor}(kw, K_{SI}, K_{SO})$: The trapdoor generation algorithm takes a keyword kw, two symmetric keys as inputs and returns a trapdoor.
- $\mathsf{id}(D(kw))$ or $\perp \leftarrow \mathsf{Match}(td, I, K, s)$: The matching algorithm takes a trapdoor, an index, a symmetric key and a secret value as inputs, and returns eid if one result is matched, otherwise returns $\perp$.

B. Details of Mechanism

In this subsection, we introduce the details of the proposed mechanism. Each entity in our network architecture invokes at least one of the algorithms mentioned above. We divide our mechanism into seven main stages: *Initialization*, *Publish and Setup*, *iO-based NIKE*, *Index Generation*, *Outsourcing*, *Search* and *Decryption*. The system view of the mechanism are shown in Fig. 2.

Initialization: To build a global environment for all users in cloud, the cloud server selects a security parameter λ and runs Init algorithm to obtain $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, p)$, where $\mathbb{G}_1, g$, and p are the global parameters for the users. $\hat{e}$ and $\mathbb{G}_2$ are held by the cloud server. Then, the cloud server randomly chooses a number s_0 as the secret value and calculates the public value g^{s_0}. For each user $u \in U$, user u selects a number s_u by random and calculates a public value g^{s_u}. The cloud server and user u exchange their public values and generate the key

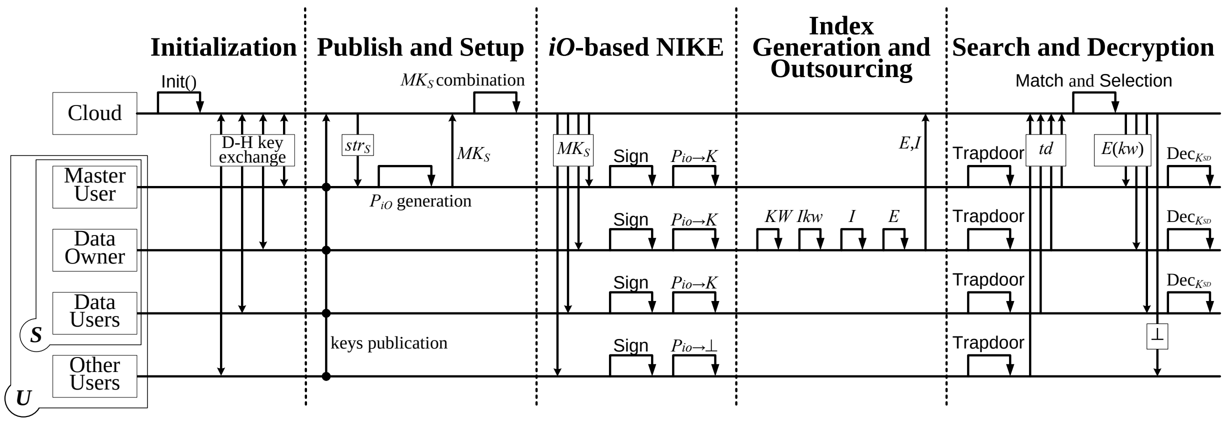

Fig. 2. A System View of the Mechanism

$K_{Cu} = (g^{s_0})^{s_u} = (g^{s_u})^{s_0}$. This protocol is called Diffie-Hellman key exchange [24]. Note that the key generated by the cloud server and the data owner is denoted as K_{CO}.

Publish and Setup: The public parameters for the authorized users are generated in this stage. This stage includes two steps: *publish* and *public parameters setup*.

In *publish* step, each user $u \in U$ invokes $\mathsf{UKeyGen}$ algorithm to obtain a key pair (sk_u, pk_u). User u holds the secret key sk_u in private and posts the public key pk_u to the cloud server. The cloud server records pk_u into T.

In *public parameters setup* step, the master user sends identifier request to the cloud server for str_S. The cloud server runs $\mathsf{Interpret}$ algorithm and returns str_S to the master user. The master user randomly selects two values to generate two instances of pseudorandom functions F_I and F_D, respectively. As show in Fig. 3 [11], F_I and F_D are the necessary components to build the programs P_{SI} and P_{SD}, respectively. Assume that the indistinguishability obfuscator $i\mathcal{O}$ is already known, the master user generates two obfuscated programs $P_{i\mathcal{O}_{SI}} = i\mathcal{O}(P_{SI})$ and $P_{i\mathcal{O}_{SD}} = i\mathcal{O}(P_{SD})$, and sends $\{str_S, P_{i\mathcal{O}_{SI}}, P_{i\mathcal{O}_{SD}}\}$ to the cloud server. The cloud server looks up T to pick out a public key set $\{pk_u\}_{u\in S}$. The cloud server then selects a new pseudorandom function f. Finally, the public parameters are denoted as $MK_S = \{str_S, P_{i\mathcal{O}_{SI}}, P_{i\mathcal{O}_{SD}}, \{pk_u\}_{u\in S}, f\}$. As the security model mentioned, MK_S is also known by the eavesdropper.

Inputs: $str_S, \{pk_u\}_{u\in S}, \sigma$
Constants: F

- If there is no $u \in S$ such that $\mathsf{DS.Ver}(pk_u, str_S, \sigma)$ passes, output $\perp$
- Otherwise, output $\mathsf{F}(str_S, \{pk_u\}_{u\in S})$

Fig. 3. the Program P_S

Non-Interactive Key Exchange In this stage, two shared symmetric keys K_{SI} and K_{SD} will be calculated by the authorized users. For any user $u \in S$, user u downloads MK_S from the cloud server and invokes $\mathsf{SKeyGen}$ algorithm to generate the shared symmetric keys. Specifically, user u computes the signature $\sigma_u = \mathsf{DS.Sign}(sk_u, str_S)$, and invokes the obfuscated programs $P_{i\mathcal{O}_{SI}}(str_S, \{pk_u\}_{u\in S}, \sigma_u)$ and $P_{i\mathcal{O}_{SD}}(str_S, \{pk_u\}_{u\in S}, \sigma_u)$ to obtain the shared symmetric keys K_{SI} and K_{SD}, respectively. During the whole process of generating the shared symmetric keys, no interaction among the authorized users is required. To generate the shared symmetric keys, each authorized user only needs to download the public parameters MK_S from the cloud server. Note that the unauthorized users can also download MK_S and try to generate the shared symmetric keys. But without holding any valid secret key, the unauthorized user fail to pass $\mathsf{DS.Ver}$ in both $P_{i\mathcal{O}_{SI}}$ and $P_{i\mathcal{O}_{SI}}$. If an unauthorized user tries to pass $\mathsf{DS.Ver}$, it has to replace one public key in $\{pk_u\}_{u\in S}$ with its own public key and to use its own secret key to generate a signature. If so, the keys generated by this unauthorized user will not be equal to the keys generated by the authorized users.

Index Generation: After the shared symmetric keys are calculated, the data owner invokes $\mathsf{IndexEnc}(D, K_{SI}, K_{CO})$ to generate a secure index I for D. Concretely, the data owner scans D and generates a keyword set $KW = \{kw_j\}_{1\le j\le num_{KW}}$. For security concern, each keyword kw_j in KW is encrypted into $ekw_j = f_{K_{SI}}(kw_j)$, where $f_{K_{SI}}$ is an instance of the public pseudorandom function f for the key K_{SI}. For each keyword $kw_j \in KW$, the data owner also generates the identifier set id$(D(kw_j))$ which includes the identifiers of the documents that contain the keyword kw_j. Then, the data owner executes $eid_j = \mathsf{SKE.Enc}(K_{CO}$,id$(D(kw_j)))$ to encrypt id$(D(kw_j))$. Thus, the secure index I can be constructed as $I = \{Ikw_j, eid_j\}_{1\le j\le num_{KW}}$, where $Ikw_j = g^{ekw_j} \cdot K_{CO}$. Fig. 4 illustrates an example of I for a D including five keywords and four documents.

Outsourcing: The data owner encrypts each $D_i \in D$ through running E_i =$\mathsf{SKE.Enc}(K_{SD}, D_i)$, and outsources the secure index I together with $\{$id(D_i),$E_i\}_{1\le i\le num_D}$ to the cloud server.

Search: As illustrated in Fig. 5, this stage includes three steps: *trapdoor generation*, *trapdoor matching* and *documents*

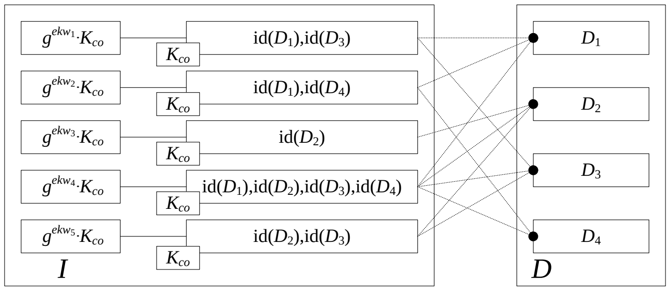

Fig. 4. An example of index I

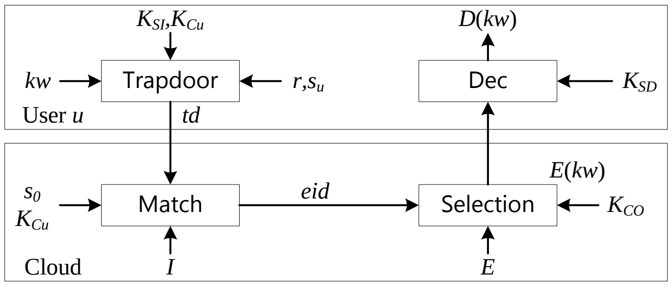

Fig. 5. The process of search and decryption

selection. Assume that an authorized user u intends to search over the encrypted documents for a keyword kw. And the keyword kw can perfectly match a corresponding keyword in the index I. Firstly, user u downloads the public parameters MK_S and calculates K_{SI} and K_{SD}.

In *trapdoor generation* step, the authorized user u runs Trapdoor to generate a trapdoor td. Specifically, user u generates the encrypted keyword $ekw = f_{K_{SI}}(kw)$ and chooses a value r by random. Then user u calculates two components $t_1 = g^{ekw \cdot r}$ and $t_2 = (K_{Cu})^r$. Finally, user u sends the trapdoor $td = \{t_1, t_2\}$ to the cloud server. The random value r ensures that the trapdoors for the same keyword are indistinguishable.

In *trapdoor matching* step, after receiving $td = \{t_1, t_2\}$ from user u, the cloud server picks out K_{Cu} and runs Match algorithm to find out the encrypted identifiers of the documents which involve the keyword kw. Specifically, for each Ikw_j in I, where $1 \le j \le num_{KW}$, the cloud server calculates the following equation. If Equation (1) holds for a certain Ikw_j, the cloud server outputs the corresponding eid_j.

$$\hat{e}(t_1, K_{Cu}) = \hat{e}(t_2, \frac{Ikw_j}{K_{CO}}) \quad (1)$$

In *documents selection* step, the cloud server runs SKE.Dec(K_{CO}, eid) to obtain id$(D(kw))$ from eid. Then the cloud server selects the corresponding $E(kw)$ and replies $E(kw)$ to user u.

Decryption: After receiving the encrypted documents set $E(kw)$, user u runs SKE.Dec with K_{SD} to decrypt every document in $E(kw)$, obtaining the documents set $D(kw)$.

VI. Security Analysis

In this section, we analyze the security of the proposed mechanism. As mentioned above, we consider an adversary $\mathcal{A}$ who can eavesdrop on all the transmission channels. Adversary $\mathcal{A}$ is a valid cloud user, but is not authorized ($\mathcal{A} \notin S$). The intention of adversary $\mathcal{A}$ is to crack the key exchange protocol so as to search over the encrypted documents without the permission of the data owner. In Theorem 1, we prove that the $i\mathcal{O}$-based NIKE is secure in our mechanism. In particular, no adversary $\mathcal{A} \notin S$ can compute the shared symmetric keys with a non-negligible probability. We then prove that $\{K_{Cu}\}_{1 \le u \le num_U}$ and index I are both secure by Theorem 2 and Theorem 3, respectively. Theorem 4 shows that $\mathcal{A}$ cannot search over the encrypted documents without the permission of the data owner.

Theorem 1. *No PPT adversary $\mathcal{A}$ ($\mathcal{A} \notin S$) can compute the symmetric key shared by users in S with non-negligible probability, provided that the following three conditions hold. (1) A constrained pseudorandom function is secure for circuit predicates; (2) $i\mathcal{O}$ is a secure indistinguishability obfuscator; (3) the constrained signature is secure.*

Proof: We prove this theorem by constructing a sequence of games, namely Game 0, Game 1, and Game 2, based on [11]. The scenario of Game 0 is equivalent to the security assumptions in the proposed mechanism. We then prove that the advantage for any adversary to win each game is equivalent. Finally, an adversary, who is able to win Game 2 with non-negligible probability, will break the security of a constrained pseudorandom function, which is assumed to be secure.

Game 0. This is an attack game based on CKS model in [25]. We simplify the CKS model to satisfy the security assumptions in our mechanism. We consider a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Assume that $\mathcal{A}$ commits to a set S which it plans to challenge. Then, $\mathcal{A}$ gets the obfuscation of P_S and makes several queries as follows:

- Register honest user queries (RegH): The challenger $\mathcal{C}$ records a tuple $(u, sk_u, pk_u, honest)$, where $u \in U$ presents one user, (sk_u, pk_u) is a key pair generated from UKeyGen and *honest* is the identity of the user. Then $\mathcal{C}$ returns pk_u to $\mathcal{A}$. This query indicates that $\mathcal{A}$ can obtain the public keys of all users in the cloud. This satisfies the assumption in our mechanism that $\mathcal{A}$ can look up T and obtain the public keys of the users.
- Register threat user queries (RegT): $\mathcal{A}$ calculates a key pair and sends public key to $\mathcal{C}$. $\mathcal{C}$ records the tuple $(u, \perp, pk_u, threat)$. In our mechanism, the user marked as *threat* is $\mathcal{A}$, who cannot be in S.
- Test queries (Test): $\mathcal{A}$ sends a challenge S to $\mathcal{C}$. $\mathcal{C}$ throws a random coin $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ picks up a secret key sk from an honest user in S, runs SKeyGen for K_S and returns K_S to $\mathcal{A}$. If $b = 1$, $\mathcal{C}$ generates a random key and returns this key to $\mathcal{A}$. Briefly, $\mathcal{C}$ returns a valid K_S to $\mathcal{A}$ if $b = 0$. Otherwise, $\mathcal{C}$ returns a random key.

After receiving the key, $\mathcal{A}$ outputs $\hat{b}$ and wins the game if

$\hat{b} = b$. It is worth noting that $\mathcal{A}$ should commit to a set S at first. Moreover, $\mathcal{A}$ can only run Test on this set. This notion is called static security [11]. Game 0 simulates the behaviors of the adversary in our mechanism. Assume that $\mathcal{A}$ can win this game with a non-negligible advantage ϵ.

Game 1. This game is identical to Game 0 except for the following modification. We construct a constrained signature with a circuit C_1 which takes a $str_{S'}$ to delegate a set S'. C_1 outputs 1 if and only if $S' \nsubseteq S$. Then, for each user $u \in S$, the challenger $\mathcal{C}$ runs $\mathsf{ConstrainGen}$ to generate a constrained key pair $(sk_{C_1 u}, pk_{C_1 u})$. When $\mathcal{A}$ queries on any user $u \notin S$, the response of RegH is the same as Game 0. When $\mathcal{A}$ queries on the user $u \in S$, the response of RegH is $pk_{C_1 u}$ instead of pk_u.

For an adversary $\mathcal{A}$, the behaviors of Game 0 and Game 1 are the same except for RegH. If $\mathcal{A}$ queries on the user u not in S, Both Game 0 and Game 1 reply the public key pk_u. Game 1 is equivalent to Game 0. If $\mathcal{A}$ queries on user u in S, Game 0 returns pk_u and Game 1 returns the constrained public key $pk_{C_1 u}$.

For adversary $\mathcal{A}$, the probability to distinguish Game 1 from Game 0 is equivalent to distinguishing $pk_{C_1 u}$ from pk_u. The probability for $\mathcal{A}$ to distinguish Game 1 from Game 0 must be negligible. If not, $\mathcal{A}$ has non-negligible advantages to breaks the security of constrained signature, which is assumed to be secure. The advantage for $\mathcal{A}$ to win Game 1 is $\epsilon - negl$.

Game 2. This game is identical to Game 1 except the following modification. In Game 1, every public key pk_u, where $u \in S$, is replaced by an constrained public key $pk_{C_1 u}$. According to the definition of C_1, $pk_{C_1 u}$ has no valid signature related to S. This is equivalent to that F in P_S will never be executed when P_S takes $S' \subseteq S$ and $\{pk_u\}_{u \in S'}$ as the inputs. Now we construct a circuit C_2 which takes $str_{S'}$ and a set of public key $\{pk_u\}_{u \in S'}$ as inputs. C_2 outputs 1 if and only if $S' \nsubseteq S$. Then we generate the constrained function F^{C_2} and construct P'_S in Fig. 6.

Inputs: $str_S, \{pk_u\}_{u \in S}, \sigma$
Constants: F^{C_2}

- If there is no $u \in S$ such that $\mathsf{DS.Ver}(pk_u, str_S, \sigma)$ passes, output $\perp$
- Otherwise, output $\mathsf{F}^{C_2}(str_S, \{pk_u\}_{u \in S})$

Fig. 6. the Program P'_S

Because the constrained public keys $\{pk_u\}_{u \in S}$ have no valid signature for S, the program P_S in Game 1 and P'_S in Game 2 perform the same functionality. Due to the definition of $i\mathcal{O}$, the obfuscation of P_S and the obfuscation of P'_S are indistinguishable. The only difference between Game 2 and Game 1 is that the obfuscation of P_S is replaced by the obfuscation of P'_S, which indicates that Game 2 is equivalent to Game 1. The advantage for $\mathcal{A}$ to break Game 2 is $\epsilon - negl$.

If there exists an adversary $\mathcal{A}$ who is able to win Game 2 with non-negligible probability, then we can use $\mathcal{A}$ to construct a new adversary $\mathcal{B}$ to break the security of the constrained pseudorandom function. After $\mathcal{A}$ commits to S, $\mathcal{B}$ requests F^{C_2} from $\mathcal{C}$, builds the program P'_S and sends $P'_{i\mathcal{O}_S} = i\mathcal{O}(P'_S)$ to $\mathcal{A}$. When $\mathcal{A}$ runs RegH, $\mathcal{B}$ responds the public key generated in Game 1. For RegT, $\mathcal{B}$ just records the public key from $\mathcal{A}$. In Test phase, $\mathcal{B}$ makes a challenge for F and receives one key from $\mathcal{C}$. Then $\mathcal{B}$ sends the key to $\mathcal{A}$. For $\mathcal{A}$, this adversary $\mathcal{B}$ perfectly plays the role as the challenger $\mathcal{C}$ in Game 2. It indicates that the probability for $\mathcal{B}$ to distinguish the key is equivalent to the probability for $\mathcal{A}$ to win Game 2. Thus, $\mathcal{B}$ breaks the security of constrained F with non-negligible advantage $\epsilon - negl$, which contradicts the assumption that the constrained pseudorandom function F is secure. The advantage for $\mathcal{A}$ to break Game 2 must be negligible, same as in Game 1 and Game 0. Proof is completed. ■

Theorem 2. *The key K_{Cu} is secure if the CDH problem is hard in $\mathbb{G}_1$.*

Proof: As mentioned above, K_{Cu} is only shared between user u and the cloud server. The adversary $\mathcal{A}$ intercepts g^{s_0} and g^{s_u} when the cloud server and user u run Diffie-Hellman key exchange in **Initialization** stage. If $\mathcal{A}$ calculates $K_{Cu} = g^{s_0 s_u}$ with non-negligible probability, it means that $\mathcal{A}$ found a way to solve the CDH problem with non-negligible probability. This contradicts the CDH assumption. $\mathcal{A}$ cannot calculate K_{Cu}. K_{Cu} is secure. Proof is completed. ■

According to Theorem 1, $\mathcal{A}$ cannot obtain K_{SI} and K_{SD} with non-negligible advantage. It proves that the shared symmetric keys are secure in our mechanism. Theorem 2 proves that $\mathcal{A}$ cannot generate any key in $\{K_{Cu}\}_{1 \le u \le num_U}$ with non-negligible probability.

Theorem 3. *No PPT adversary $\mathcal{A}$ can extract any keyword from index I with non-negligible probability, provided that the pseudorandom function and the symmetric key encryption are both secure.*

Proof: According to Theorem 2, it is hard for $\mathcal{A}$ to generate K_{CO}. Due to the assumption that the SKE is secure, $\mathcal{A}$ cannot decrypt any eid without owning K_{CO} when I is intercepted. Therefore, the proof of Theorem 3 is equivalent to proving that the index I is secure.

Assume that there is an adversary $\mathcal{A}$ who can get the keyword kw from I. Because only ekw involves kw in I, $\mathcal{A}$ must find a solution to obtain kw from $f_{K_{SI}}(kw)$. According to the security of pseudorandom function, $\mathcal{A}$ cannot distinguish $f_{K_{SI}}$ from any random function. If $\mathcal{A}$ can obtain kw from $f_{K_{SI}}(kw)$, it is equivalent to breaking the security of pseudorandom function. That contradicts the assumption that the pseudorandom function is secure. Proof is completed. ■

Theorem 4. *The probability of successful searching is negligible for any PPT adversary $\mathcal{A}$, provided that the DL problem is hard and that the pseudorandom function is secure.*

Proof: As we assumed before, $\mathcal{A}$ plays two different roles: an eavesdropper and an unauthorized user. We consider that $\mathcal{A}$ attempts to search over the encrypted documents via directly using the intercepted trapdoors from the authorized users or forging some trapdoors.

Case 1. If $\mathcal{A}$ plays the role of an eavesdropper, $\mathcal{A}$ can intercept the trapdoors sent from the authorized users to the cloud server. Assume that $\mathcal{A}$ intercepts a trapdoor td from user u and directly sends td to the cloud server as its own search request. Note that K_{CA} is the key generated by $\mathcal{A}$ and the cloud server in **Initialization** stage. After receiving td, the cloud server runs Match algorithm and calculates both sides of Equation 1 for all $Ikw \in I$.

$$\hat{e}(t_1, K_{CA}) = \hat{e}(g^{ekw\cdot r}, g^{s_0\cdot s_A}) = \hat{e}(g,g)^{ekw\cdot r\cdot s_0\cdot s_A}$$
$$\hat{e}(t_2, \frac{Ikw}{K_{CO}}) = \hat{e}((K_{Ci})^r, g^{ekw}) = \hat{e}(g,g)^{ekw\cdot r\cdot s_0\cdot s_i}$$

The above equations show that if the keyword in td matches one keyword in I, Equation 1 holds only when $\mathcal{A}$ can calculate a secret value s_A such that $s_A = s_i$. Because g and g^{s_i} are known by $\mathcal{A}$, the solution of DL problem can be found if $\mathcal{A}$ can generate a secret value s_A where $s_A = s_i$. Due to the hardness of solving DL problem, $\mathcal{A}$ cannot generate such s_A, where $s_A = s_i$, with non-negligible probability. $\mathcal{A}$ cannot directly use trapdoors from the authorized users to complete the search successfully.

Case 2. If $\mathcal{A}$ plays the role of an unauthorized user, $\mathcal{A}$ intends to forge a trapdoor for keyword search. According to Equation 1, $\mathcal{A}$ can find a match in Match algorithm only if it can calculate a ekw to match one ekw in I. Theorem 1 proves that $\mathcal{A}$ cannot obtain K_{SI}. As long as the pseudorandom function is secure, $\mathcal{A}$ cannot forge a valid ekw without knowing K_{SI}.

In neither case, $\mathcal{A}$ can search over the encrypted documents. This concludes the proof. ∎

VII. Conclusion

In this paper, we have proposed a keyword search mechanism with multi-user setting under a more stringent scenario where no secure transmission channel exists between the data owner and the data user. Compared with the existing mechanisms, our mechanism provides secure keyword search over encrypted data even in the presence of a stronger adversary who can eavesdrop on all transmission channels. Our mechanism does not require any authentication authority for access control. While all users can perform search in the mechanism, only those authorized users will get correct results. Our analysis shows the following security properties of our mechanism. 1) The shared symmetric keys are secure in the mechanism and cannot be calculated with non-negligible probability by the eavesdropper; 2) the index is secure, from which the eavesdropper cannot extract any information about the keyword; 3) no authentication authority is needed in the mechanism – unauthorized users cannot successfully search over the shared documents, neither by forging the trapdoors nor using the intercepted trapdoors directly.

References

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 136–149.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000, pp. 44–55.

[3] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology-Eurocrypt 2004*. Springer, 2004, pp. 506–522.

[4] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of ACM conference on Computer and communications security*, 2006, pp. 79–88.

[5] 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*. Springer, 2008, pp. 71–85.

[6] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. of IEEE ICDCS*, 2010.

[7] Y. Yang, H. Lu, and J. Weng, "Multi-user private keyword search for cloud computing," in *Proc. of IEEE Cloud Computing Technology and Science*, 2011.

[8] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Computational Science and Its Applications–ICCSA 2008*. Springer, 2008, pp. 1249–1259.

[9] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. of IEEE INFOCOM*, 2014.

[10] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. of IEEE INFOCOM*, 2014.

[11] D. Boneh and M. Zhandry, "Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation," in *Advances in Cryptology–CRYPTO 2014*. Springer, 2014, pp. 480–499.

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

[13] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security*. Springer, 2004, pp. 31–45.

[14] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Applied Cryptography and Network Security*. Springer, 2005, pp. 442–455.

[15] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Proc. of IEEE ICDCS Workshops*, 2011.

[16] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. of ACM conference on Computer and communications security*, 2012.

[17] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. of IEEE INFOCOM*, 2010.

[18] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. of IEEE INFOCOM*, 2012.

[19] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1467–1479, 2012.

[20] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. of IEEE INFOCOM*, 2014.

[21] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 222–233, 2014.

[22] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proc. of IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2013.

[23] D. Boneh and B. Waters, "Constrained pseudorandom functions and their applications," in *Advances in Cryptology-ASIACRYPT 2013*. Springer, 2013, pp. 280–300.

[24] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.

[25] D. Cash, E. Kiltz, and V. Shoup, "The twin diffie-hellman problem and applications," in *Advances in cryptology–EUROCRYPT 2008*. Springer, 2008, pp. 127–145.