2016

Threshold broadcast encryption with keyword search

Shiwei Zhang
University of Wollongong, sz653@uowmail.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Guomin Yang
University of Wollongong, gyang@uow.edu.au

Publication Details

Zhang, S., Mu, Y. & Yang, G. (2016). Threshold broadcast encryption with keyword search. Lecture Notes in Computer Science, 9589 322-337. Beijing, China Revised Selected Papers of the 11th International Conference, Inscrypt 2015

Threshold broadcast encryption with keyword search

Abstract
Many users store their data in a cloud, which might not be fully trusted, for the purpose of convenient data access and sharing. For efficiently accessing the stored data, keyword search can be performed by the cloud server remotely with a single query from the user. However, the cloud server cannot directly search the data if it is encrypted. One of solutions could be to allow the user to download the encrypted data, in order to carry out a search; however, it might consume huge network bandwidth. To solve this problem, the notion of keyword search on encrypted data (searchable encryption) has been proposed. In this paper, a special variant of searchable encryption with threshold access is studied. Unlike some previous proposals which have fixed group and fixed threshold value, we define a new notion named Threshold Broadcast Encryption with Keyword Search (TBEKS) for dynamic groups and flexible threshold values. We formalize the security of a TBEKS scheme via a new security model named IND-T-CKA which captures indistinguishability against chosen keyword attacks in the threshold setting. We also propose the first practical TBEKS scheme with provable security in our IND-T-CKA security model, assuming the hardness of the Decisional Bilinear Diffie-Hellman problem.

Keywords
encryption, keyword, threshold, search, broadcast

Disciplines
Engineering | Science and Technology Studies

Publication Details
Zhang, S., Mu, Y. & Yang, G. (2016). Threshold broadcast encryption with keyword search. Lecture Notes in Computer Science, 9589 322-337. Beijing, China Revised Selected Papers of the 11th International Conference, Inscrypt 2015

Threshold Broadcast Encryption with Keyword Search

Shiwei Zhang, Yi Mu, and Guomin Yang

Centre for Computer and Information Security Research
School of Computing and Information Technology
University of Wollongong, Australia
{sz653,ymu,gyang}@uow.edu.au

Abstract. Many users store their data in a cloud, which might not be fully trusted, for the purpose of convenient data access and sharing. For efficiently accessing the stored data, keyword search can be performed by the cloud server remotely with a single query from the user. However, the cloud server cannot directly search the data if it is encrypted. One of solutions could be to allow the user to download the encrypted data, in order to carry out a search; however, it might consume huge network bandwidth. To solve this problem, the notion of keyword search on encrypted data (searchable encryption) has been proposed. In this paper, a special variant of searchable encryption with threshold access is studied. Unlike some previous proposals which have fixed group and fixed threshold value, we define a new notion named *Threshold Broadcast Encryption with Keyword Search* (TBEKS) for dynamic groups and flexible threshold values. We formalize the security of a TBEKS scheme via a new security model named IND-T-CKA which captures indistinguishability against chosen keyword attacks in the threshold setting. We also propose the first practical TBEKS scheme with provable security in our IND-T-CKA security model, assuming the hardness of the Decisional Bilinear Diffie-Hellman problem.

Keywords: Searchable Encryption, Keyword Search, Cloud Security

1 Introduction

Cloud computing [8] provides flexible computing resources, including data storage, to end users. Users are able to upload their data to the cloud for later access by themselves or by other users (i.e., data sharing) via the Internet. In other words, on-demand data access is available via the Internet where users can search and then download what they need. To prevent a huge amount of network bandwidth consumption, the search operations are usually done by the cloud instead of letting users download all the data and search locally.

Meanwhile, to ensure the privacy of the users, some sensitive data should be protected against the cloud server while the keyword search functionality is maintained. Specifically, the data to be searched and the keyword used in the

search operation should be inaccessible by any non-authorised parties, including the cloud. With such a demand, various searchable encryption schemes [1, 3, 4, 7, 11] have been proposed to enable secure searching over encrypted data. In a public key searchable encryption scheme, Bob encrypts both the data and the keywords under Alice's public key and uploads the ciphertexts to the cloud. As both the data and the keywords are protected, it is hard for the cloud server to gain any information about the data. To perform search operations, Alice generates a trapdoor for a keyword [1, 3, 4] or multiple keywords [7] and transfers it to the cloud via a secure communication channel. Upon receiving Alice's trapdoor of the keyword, the cloud server searches the whole database and returns the search results back to Alice. Finally, Alice downloads the ciphertexts from the cloud based on the search results, and decrypts them to get the original data.

In the normal searchable encryption schemes, the accessibility to the data and its search operation is authorised to a user [1, 4] or a set of users [12, 15] where any *single* user in the authorised user set can perform the search and the decryption operations. However, a single identity may not be trustful in some scenarios. For instance, a research team of a company is developing a new product and needs to access the company database. The head of the research department does not trust any single member of the research team to access the database, since an individual member may leak the secrets of the company for monetary purposes. To reduce the risk of a single point failure, a threshold searchable encryption scheme is more suitable where the accessibility to the data is decentralised from a single member to n members of the team where searching the database and decrypting a ciphertext both require at least t members to work together. To be more precise, in order to perform a search operation successfully, the cloud needs to obtain for a keyword at least t trapdoors from the n authorised users. If such a threshold searchable encryption also supports dynamic groups and flexible threshold values, the company can specify different classifications for different data by changing the authorised user set and the threshold value t. This paper aims to provide a practical solution for this problem.

1.1 Related Work

Boneh et al. [4] introduced the searchable encryption, namely public encryption with keyword search (PEKS), and defined the security model that the adversary cannot identify the keyword from the ciphertext without a trapdoor. Xu et al. [14] argued that PEKS is insecure under key guessing attack (KGA) since the remote server can always create a ciphertext of a keyword and test it with the target trapdoor. If the keyword space is in polynomial size, the adversary can get the keyword from the target trapdoor in polynomial time. Xu et al. [14] also proposed a method to enhance the security under KGA by encrypting and searching for the fuzzy keyword instead of the exact keyword.

Boneh et al. [4] showed that PEKS implies identity-based encryption (IBE) but not vice versa. Nevertheless, Abdalla et al. [1] proposed a PEKS scheme generically constructed using anonymous identity-based encryption (AnonIBE). They also proposed identity-based searchable encryption (IBKS) from a 2-leveled

anonymous hierarchical identity-based encryption (AnonHIBE). Similarly, searchable broadcasting encryption, namely broadcast encryption with keyword search (BEKS), can be constructed using 2-leveled anonymous hierarchical identity-coupling broadcast encryption (AnonHICBE) [2].

Searchable encryption can be divided into the single user setting (e.g. PEKS) and the multi-user setting. Broadcast encryption with keyword search [2] and attribute-based encryption with keyword search (ABKS) [12, 15] are in the multi-user setting. In BEKS, the keyword is encrypted for a set of users. If a user is in the target set, the user can generate the trapdoor for testing the ciphertext. In ABKS, the keyword is encrypted under a policy or with attributes. Only the user who has a match of the policy and the attributes can generate the trapdoor for testing the ciphertext.

However, in both BEKS and ABKS, the individual target user has the full ability to generate the trapdoor. Wang et al. [13] decentralised the ability of trapdoor generation to multi-user in a threshold manner, which requires at least k of n users to generate the trapdoor. Siad [10] gave a formal definition of threshold public key encryption with keyword search (TPEKS), and generically constructed a TPEKS scheme with threshold (n, t)-IBE but no concrete scheme is provided. In Wang et al.'s scheme [13], a trusted centralised manager is required to generate the private keys for all users. To enhance the security, Siad's scheme [10] leverages a distributed protocol in private keys generation instead of a trusted third party.

We find that both schemes [10, 13] are limited to a fixed number of users and fixed threshold value at the key generation stage. It makes adding or removing a user impossible, and changing the threshold value for individual ciphertext impossible. To encrypt a keyword for different set of users or with different threshold value, we have to generate the private keys for all the users in the target set. If it is an (n, t)-TPEKS scheme where t is the threshold value such that $0 < t \leq n$ and n is the maximum number of users, the users have to store $O(n \cdot 2^n)$ private-public key pairs for the possible ciphertexts, although they may share the same global public parameters.

1.2 Our Contribution

In this paper, we introduce a new notion named *Threshold Broadcast Encryption with Keyword Search* (TBEKS). We provide a formal definition of TBEKS and a formal security model, named indistinguishability in the threshold setting against chosen keyword attack (IND-T-CKA), to capture its security. Moreover, we construct a practical TBEKS scheme and prove that it is IND-T-CKA secure under the Decisional Bilinear Diffle-Hellman (DBDH) assumption in random oracle model.

In our TBEKS definition and scheme, users are ad hoc, i.e., they can generate their own private-public key pair individually. The data owner selects a target set of users and threshold value t to encrypt a keyword, and then uploads the full ciphertext to the remote server. To search the files containing a certain keyword, at least t users of the target user set need to generate their trapdoor shares for

that keyword, and transfer those trapdoor shares to the remote server, in order to enable the remote server to perform the search operation. Our scheme does not fix the user group and the threshold value at the system setup, and only one private-public key pair is required for each user. Thus we solve Wang et al.'s open problem for dynamic group [13].

1.3 Paper Organisation

The rest of this paper is organised as follows. In section 2, we review some essential tools and assumptions, including threshold secret sharing schemes and bilinear maps. We define TBEKS and its security model in section 3. Then we propose our TBEKS scheme in section 4 and prove that it is secure under the security model defined in section 3.2. Finally, conclusion is addressed in section 5.

2 Preliminaries

2.1 Threshold Secret Sharing Scheme

Shamir's secret sharing scheme [9] divides a secret s into n pieces $s_1, \ldots, s_n$ using a $k-1$ degree polynomial and distributes to n users. If and only if k users or more come together, they can recover s by polynomial interpolation. Knowing $k-1$ pieces of s does not reveal any information about s. This scheme is also called (k, n) *Threshold Secret Sharing Scheme*. Details are shown as follows.

Let $GF(q)$ be a finite field with order q where $q > n$. Each user U_i is associated with a public unique number $u_i \in GF(q)$. We also represent randomly choosing r from a space $\mathbb{S}$ by $r \in_R \mathbb{S}$. To share a secret $s \in GF(q)$ among a user set $S = \{U_1, \ldots, U_n\}$, a random $k-1$ degree polynomial is picked as $p(x) = s + \sum_{j=1}^{k-1} a_j x^j$ where $a_j \in_R GF(q)$. Each user in the user set S gets a share $s_i = p(u_i)$. When k users come together and form a user set $A \subset S$, we can recover $p(x) = \sum_{U_i \in A} \Delta_i^A s_i$ where $\Delta_i^A = \prod_{U_\ell \in A \wedge i \neq \ell} \frac{x - u_\ell}{u_i - u_\ell}$. Then we can recover $s = p(0)$. Obviously, we can recover any point by

$$s_j = p(u_j) = \sum_{U_i \in A} \Delta_{ij}^A s_i \quad \text{where } \Delta_{ij}^A = \prod_{U_\ell \in A \wedge i \neq \ell} \frac{u_j - u_\ell}{u_i - u_\ell}.$$

By defining $u_0 = 0$, we have $s = \sum_{U_i \in A} \Delta_{i0}^A s_i$.

2.2 Bilinear Maps

A bilinear map is a function that maps two group spaces to a third space. For simplicity, we exploit the same bilinear map used in [5] where the first two group spaces are the same. Let $\mathbb{G}_1$ be a additive group, $\mathbb{G}_2$ be a multiplicative group, and both are cyclic groups of prime order q. Let P be a generator of $\mathbb{G}_1$. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ has the following properties:

- Bilinearity: $\forall a, b \in \mathbb{Z}_q, e(aP, bP) = e(P, P)^{ab}$.

– Non-Degeneracy: $e(P, P) \neq 1$.
– Efficiency: It can be computed for any possible input efficiently.

There is a computational hard problem named *Decisional Bilinear Diffie-Hellman problem* (DBDH) coming with the bilinear map that we can rely on to construct our cryptographic scheme. The definition of DBDH problem is shown as follows.

Definition 1 (Decisional Bilinear Diffie-Hellman problem). *Let a, b, c be uniformly and independently chosen from $\mathbb{Z}_q$, and T be uniformly and independently chosen from $\mathbb{G}_2$. Giving two probability distributions $\mathcal{D}_{BDH} = (P, aP, bP, cP, e(P, P)^{abc})$ and $\mathcal{D}_{rand} = (P, aP, bP, cP, T)$, there is an algorithm $\mathcal{A}$ can distinguish $\mathcal{D}_{BDH}$ and $\mathcal{D}_{rand}$ with advantage:*

$$Adv_{\mathcal{A}}^{DBDH} = \frac{1}{2}\left|\Pr[1 \leftarrow \mathcal{A}(D \xleftarrow{U} \mathcal{D}_{BDH})] - \Pr[1 \leftarrow \mathcal{A}(D \xleftarrow{U} \mathcal{D}_{rand})]\right|$$

where $D \xleftarrow{U} \mathcal{D}_{BDH}$ means that D is uniformly and independently chosen from $\mathcal{D}_{BDH}$. The advantage can be represented alternatively as

$$D_0 \xleftarrow{U} \mathcal{D}_{BDH}, \quad D_1 \xleftarrow{U} \mathcal{D}_{rand}, \quad b \xleftarrow{U} \{0, 1\},$$
$$Adv_{\mathcal{A}}^{DBDH} = \left|\Pr[b = b' \leftarrow \mathcal{A}(D_b)] - \frac{1}{2}\right|.$$

The DBDH problem is computational hard if and only if the advantage $Adv_{\mathcal{A}}^{DBDH}$ is negligible. In other words, it is hard to distinguish whether a vector is chosen from $\mathcal{D}_{BDH}$ or $\mathcal{D}_{rand}$ other than a random guess. Our scheme is secure based on the assumption of the hardness of the DBDH problem.

3 Threshold Broadcast Encryption with Keyword Search

3.1 Definition

Generally speaking, a *Threshold Broadcast Encryption with Keyword Search* (TBEKS)[1] scheme is used along with a *Threshold Broadcast Encryption* (TBE) scheme [6], where the former encrypts the keywords and the latter encrypts the message[2] . Independent private-public key pairs are suggested for the combination of the above mentioned system. In TBKES, there are three roles involved, including the **data owner** who encrypts the message and the keywords, the **server** who stores the ciphertexts and performs the requested search, and the **user** who has the access to the decryption of the message and generates search queries. TBEKS works as follows. The **data owner** chooses a set of **users** and a threshold value t, and encrypts the message under TBE and the keyword under TBEKS. Then the **data owner** combines the ciphertexts and uploads them to

[1] We choose the name TBEKS in order to separate it from TPEKS.
[2] In a storage system, messages are actually files.

the **server**. To perform a search operation, at least t **users** generate their individual trapdoors for the same target keyword W and upload the trapdoors to the **server** via a secure communication channel. After that, the **server** searches the whole database of ciphertexts with the given trapdoors and returns the result message indices back. Upon receiving the indices, the **users** retrieve the corresponding ciphertexts and decrypt them with at least t **users** working together. Note that only the trapdoors are required to be transferred via a secure communication channel.

Formally, we present the definition of *Threshold Broadcast Encryption with Keyword Search* as follows.

Definition 2 (Threshold Broadcast Encryption with Keyword Search). *A threshold broadcast encryption with keyword search scheme, involving the data users, the servers and the users U_i, consists of the following five possibly probabilistic polynomial time algorithms:*

- $params \leftarrow Setup(1^k)$*: The randomised system setup algorithm takes a security parameter 1^k, and outputs a set of parameters used in the system widely. This algorithm can be run by anyone whereas all users are required to agree on the same parameters.*
- $(PK_i, SK_i) \leftarrow KeyGen(params)$*: The randomised user key generation algorithm takes a system parameter $params$, and outputs a pair of secret key SK_i and public key PK_i of a user U_i. This algorithm is run by the users individually.*
- $C \leftarrow TBEKS(\{PK_1, \ldots, PK_n\}, t, W)$*: The randomised keyword encryption algorithm takes a set of public keys $\{PK_1, \ldots, PK_n\}$ of n target users, a threshold value t and a keyword W, and outputs a ciphertext C of the keyword W. This algorithm is run by the data owner.*
- $T \leftarrow Trapdoor(SK_i, W)$*: The possibly randomised trapdoor generation algorithm takes the secret key SK_i of a user U_i and a keyword W, and outputs a user trapdoor T of the keyword W. This algorithm is run by the users individually.*
- $1/0 \leftarrow Test(\{T_1, \ldots, T_t\}, C)$*: The deterministic test algorithm takes t trapdoors $T_i \leftarrow Trapdoor(SK_i, W)$ and a keyword ciphertext $C \leftarrow TBEKS(\{PK'_1, \ldots, PK'_n\}, t', W')$, and outputs*

$$\begin{cases} 1 & \text{if } W = W' \wedge t \geq t' \wedge \{PK_1, \ldots, PK_n\} \subset \{PK'_1, \ldots, PK'_n\}, \\ 0 & \text{otherwise.} \end{cases}$$

where $(PK_i, SK_i) \leftarrow KeyGen(params)$. This algorithm is run by the servers and the results will be sent back to each user involved.

In addition, we require the scheme to be correct.

Definition 3 (Correctness). *A threshold broadcast encryption with keyword search scheme is correct if the following statement is always true:*

$$\forall params \leftarrow Setup(1^k), \quad \forall (SK, PK) \leftarrow KeyGen(params),$$
$$\forall n, t \in \mathbb{Z}^+ \wedge t \leq n, \quad \forall W \in \{0,1\}^*, \quad \forall C \leftarrow TBEKS(\{PK_1, \ldots, PK_n\}, t, W),$$
$$\forall S \subset \{1, \ldots, n\} \wedge t \leq |S| \leq n,$$
$$Test(\{T \mid T \leftarrow Trapdoor(SK_i, W) \wedge i \in S\}, C) = 1.$$

3.2 Security Model

In Definition 2, we implicitly allow the server to combine the trapdoors for a keyword freely without any interaction with related users. For instance, the data owner creates $C_1 \leftarrow TBEKS(\{PK_1, PK_2\}, 2, W)$. To search for C_1, the users U_1, U_2 generate $T_i \leftarrow Trapdoor(SK_i, W)$ for $i = 1, 2$. Later, the data owner creates $C_2 \leftarrow TBEKS(\{PK_2, PK_3\}, 2, W)$ for the same keyword W. Similarly, to search for C_2, the users U_2, U_3 generate $T'_i \leftarrow Trapdoor(SK_i, W)$ for $i = 2, 3$. If $Trapdoor$ is a deterministic algorithm, the server can easily link T_1, T_2 and T'_3 together that they are created for the same keyword since $T_2 = T'_2$. As a result, if the data owner creates $C_3 \leftarrow TBEKS(\{PK_1, PK_2, PK_3\}, 3, W)$, the server can search C_3 by $Test(\{T_1, T_2, T'_3\}, C_3) = 1$. However, the server gains no information, especially the keyword encrypted in the trapdoors, other than the test result. Instead, this provides a feature that the server can cache the uploaded trapdoors from the users.

Because of this feature, we consider that the server is *honest but curious*. Importantly, we do not allow the server to collude with any users. Otherwise, the user U_1 and the server can learn the keyword in the ciphertext. For example, the server gets T_1, T_2 and T_3 from the users to test C_3. Then the server can use T_2 and T_3 to test C_2, and return the result to the user U_1. Now, the user U_1 knows the keyword of a ciphertext while U_1 is not in the target user set.

We also do not consider the keyword guessing attack (KGA) [14], since the server can always create a ciphertext $C = TBEKS(\{PK\}, 1, W)$ for all the keywords W with the user's trapdoor T. Commonly, if the keyword space is polynomial sized, the server can get the corresponding keyword W of C in polynomial time. However, this kind of attack can be prevented by Xu et al.'s method [14]. The BDOP scheme [4] also does not consider this attack.

In threshold broadcast encryption with keyword schemes, many users are involved. We consider that all users are registered before creating the ciphertexts, as the adversary may be able to register the private-public key pair of the target user. Now we define the *indistinguishability in the threshold setting against chosen keyword attack* (IND-T-CKA) game Game 1 where an active adversary $\mathcal{A}$ tries to distinguish two encryptions of keywords W_0 and W_1 with the security parameter k:

1. The challenger runs the $Setup(1^k)$ algorithm to generate a set of system-wide parameters and passes them to the adversary $\mathcal{A}$.

$Game^{k}_{IND-T-CKA}$:

$\mathcal{U}, \mathcal{C}, \mathcal{W} \leftarrow \emptyset$

$params \leftarrow Setup(1^k)$

$(\mathcal{S}, t, W_0, W_1) \leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}, \mathcal{O}_{Corrupt}, \mathcal{O}_{Trapdoor}}(params)$

$b \in_R \{0, 1\}$

$C \leftarrow TBEKS(\{PK_i\}_{i \in \mathcal{S}}, t, W_b)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{KeyGen}, \mathcal{O}_{Corrupt}, \mathcal{O}_{Trapdoor}}(C)$

$\mathcal{O}_{KeyGen}$:

$(PK_i, SK_i) \leftarrow KeyGen(params)$

$\mathcal{U} \leftarrow \mathcal{U} \cup \{U_i\}$

return PK_i

$\mathcal{O}_{Corrupt}$:

$\mathcal{C} \leftarrow \mathcal{C} \cup \{U_i\} \subset \mathcal{U}$

return SK_i

$\mathcal{O}_{Trapdoor}$:

$T \leftarrow Trapdoor(SK_i, W)$

$\mathcal{W} \leftarrow \mathcal{W} \cup \{W\}$

return T

$$Adv^{IND-T-CKA}_{\mathcal{A}} = \left| \Pr\left[b = b' \wedge |\mathcal{S} \cap \mathcal{C}| < t \wedge W_0, W_1 \notin \mathcal{W} \right] - \frac{1}{2} \right|$$

Game 1: IND-T-CKA

2. The adversary can adaptively ask the challenger to register a user and obtain the public key of that user by querying the key generation oracle $\mathcal{O}_{KeyGen}$. At the same time, the challenger records the requested user U_i in the user list $\mathcal{U}$.
3. The adversary can adaptively ask the challenger to obtain the secret key of a registered user $U_i \in \mathcal{U}$ by querying the collusion oracle $\mathcal{O}_{Corrupt}$. At the same time, the challenger records the requested user U_i in the collusion list $\mathcal{C}$.
4. The adversary can adaptively ask the challenger to obtain the user U_i's trapdoor of a keyword W by querying the trapdoor generation oracle $\mathcal{O}_{Trapdoor}$. At the same time, the challenger records the requested keyword W in the keyword list $\mathcal{W}$. For the corrupted users $U_i \in \mathcal{C}$, the adversary can compute the trapdoor by itself using the users U_i's secret key SK_i. Hence, the keyword list $\mathcal{W}$ only contains the requested keywords of uncorrupted users.
5. At some point, the adversary $\mathcal{A}$ outputs a set $\mathcal{S}$ of users, a threshold value t and two keywords W_0 and W_1 to be challenged. The adversary is restricted that W_0 and W_1 are not in the list $\mathcal{W}$ as they are not queried to $\mathcal{O}_{Trapdoor}$.

The adversary is also restricted that it cannot corrupt t users or more in the user set $\mathcal{S}$.

6. The challenger randomly selects b to be 0 or 1, and gives a ciphertext $C = TBEKS(\{PK_i\}_{i\in\mathcal{S}}, t, W_b)$ to the adversary $\mathcal{A}$.
7. The adversary can continue to query all three oracles $\mathcal{O}_{KeyGen}$, $\mathcal{O}_{Corrupt}$, $\mathcal{O}_{Trapdoor}$ with the same restrictions.
8. Eventually, the adversary $\mathcal{A}$ outputs a bit b'. If $b = b'$, the adversary wins the game.

We define the advantage of winning Game 1 as

$$Adv_{\mathcal{A}}^{IND-T-CKA} = \left| \Pr\left[b = b' \wedge |\mathcal{S} \cap \mathcal{C}| < t \wedge W_0, W_1 \notin \mathcal{W}\right] - \frac{1}{2} \right|.$$

Definition 4 (IND-T-CKA Security). *A threshold broadcast encryption with keyword search (TBEKS) scheme is indistinguishable in the threshold setting against chosen keyword attack (IND-T-CKA) if $Adv_{\mathcal{A}}^{IND-T-CKA}$ is a negligible function for all adversary $\mathcal{A}$ winning the Game 1 in polynomial time.*

4 Construction

4.1 The Scheme

We build our TBEKS scheme based on Daza et al.'s TBE scheme [6] using the idea similar to Boneh et al.'s PEKS scheme [4]. The main idea of the construction is to use the secret keys of users as the shares of a shared secret in an $(n, 2n-t)$ threshold secret sharing scheme. The shared secret works as the secret key of a dummy user in [4]. Since all computations are done with points on the elliptic curve and due to the hardness of discrete logarithm problem (DLP), the secret shares are computationally secure.

Our TBEKS scheme works as follows.

- $params \leftarrow Setup(1^k)$: Given a security parameter 1^k, this algorithm generates a prime number of q bits and groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q where there is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. This algorithm also picks a random generator P of $\mathbb{G}_1$. After that, the algorithm picks two hash functions $H : \{0,1\}^* \to \mathbb{G}_1$ and $H' : G_1 \to \mathbb{Z}_q$. Note that the hash function H is used to hash a keyword W into a point on the elliptic curve and the hash function H' is used to hash the public key PK_i of a user U_i to an domain input $u_i = H'(PK_i)$ used in the threshold secret sharing scheme. Hence, it is required H' to be a collision resistant hash function. Alternatively, instead of using a hash function H', a user U_i can select its own unique u_i and then register along with its public key PK_i to a certification authority. Thus each user U_i has a unique public key PK_i associated with a unique public value u_i. For the system simplicity, we use the hash function H'.

$$\mathbb{G}_1 = \langle P \rangle, \quad e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2, \quad H : \{0,1\}^* \to \mathbb{G}_1, \quad H' : G_1 \to \mathbb{Z}_q.$$

return $params = (q, \mathbb{G}_1, \mathbb{G}_2, P, e, H, H')$.

– $(PK_i, SK_i) \leftarrow KeyGen(params)$: With the system wide parameters $param$, each user U_i randomly chooses a secret key $SK_i = x_i \in_R \mathbb{Z}_q^+$. Then its public key can be computed as $PK_i = x_i P$.

$$SK_i = x_i \in \mathbb{Z}_q^+, \quad PK_i = x_i P.$$

return (PK_i, SK_i).

– $C \leftarrow TBEKS(\{PK_1, \ldots, PK_n\}, t, W)$: To encrypt a keyword W, the data owner first obtains all the public keys of the target users $S = \{U_1, \ldots, U_n\}$. Then the data owner obtains the associated input values $\mathcal{U} = \{u_i = H'(PK_i) \mid U_i \in S\}$. Having n input values u_i and $PK_i = x_i P$, we can recover/construct a polynomial $p(u_j)P = \sum_{U_i \in S} \Delta_{ij}^S PK_i$. To form an $(n, 2n-t)$ threshold secret sharing scheme, the data owner chooses $n-t$ unique domain input values $\mathcal{D} \leftarrow \mathbb{Z}_q^{n-t}$, where $\mathcal{U} \cap \mathcal{D} = \emptyset$, for $n-t$ dummy users. Given the $n-t$ dummy users as a dummy user set D, the data owner can compute the public keys of the dummy users by computing $PK_j = p(u_j)P$ for all $u_j \in \mathcal{D}$. The detail of this algorithm works as follows.

$$Q = p(0)P = \sum_{U_i \in S} \Delta_{i0}^S PK_i, \quad s \in_R \mathbb{Z}_q^*, \quad C_1 = sP, \quad C_2 = e(H(W), Q)^s,$$

For each dummy user $U_j \in D$,

$$PK_j = p(u_j)P = \sum_{U_i \in S} \Delta_{ij}^S PK_i, \quad K_j = e(sH(W), PK_j).$$

return $C = (S, t, D, C_1, C_2, \{K_j\}_{U_j \in D})$.
To improve computational efficiency, it is possible to reuse Q and PK_j for the same user set S and different keywords, since these two variables are irrelevant to the keyword W and the randomness s. When calculating K_j, we can calculate $sH(W)$ before the loop so that all we need is a pairing operation. For the ciphertext C, it is not necessary to include all the public keys and the associated domain input values for both real users and dummy users since we only need the domain input values later. For better efficiency, this algorithm can return the ciphertext as $C = (\mathcal{U}, t, \mathcal{D}, C_1, C_2, \{K_j\}_{U_j \in D})$. As the values in $\mathcal{D}$ are not required to be chosen uniformly, the data owner can choose a continuous interval that $\mathcal{D} = \{r, r+1, r+2, \ldots, r+n-t-1\}$ where $r \in_R \mathbb{Z}_q$. Thus $\mathcal{D}$ can be represented in two numbers in $\mathbb{Z}_q$. Hence, for the best result, the ciphertext size is $(n+3)\mathbb{Z}_q + (n-t+2)\mathbb{G}_1$.

– $T \leftarrow Trapdoor(SK_i, W)$: The user U_i generates the trapdoor T for the keyword W simply using its secret key. Then the user U_i uploads the trapdoor to the server via a secure communication channel.

$$T = x_i H(W).$$

return T.

– $1/0 \leftarrow Test(\{T_1, \ldots, T_t\}, C)$: Upon receiving t trapdoors from the users $A = \{U_1, \ldots, U_t\}$ where $|A \subset S| = t$, the server can run the following algorithm. If more than t target trapdoors are uploaded, the server only picks the first t trapdoors.

$$\text{For each user } U_i \in A, \quad K_i = e(T_i, C_1),$$
$$B = A \cup D, \quad K = \prod_{U_i \in B} K_i^{\Delta_{i0}^B}.$$

return $K \stackrel{?}{=} C_2$.

Theorem 1. *The proposed threshold broadcast encryption with keyword search scheme is correct.*

Proof. Correctness is verified as following. First, K_i can be calculated as

$$K_i = e(sH(W), PK_i) = e(sH(W), x_i P) = e(x_i H(W), sP) = e(T_i, C_1).$$

Then we continue to verify the correctness of K,

$$K = \prod_{U_i \in B} K_i^{\Delta_{i0}^B} = \prod_{U_i \in B} e(sH(W), PK_i)^{\Delta_{i0}^B} = e(H(W), \sum_{U_i \in B} \Delta_{i0}^B PK_i)^s.$$

Since the $(n, 2n-t)$ threshold secret sharing scheme is constructed by n real users S, distributing shares to $n-t$ dummy users D, any n users in $S \cup D$ can recover the polynomial p used in the $TBEKS$ algorithm. Having $|D| = n - t$ and $S \cap D = \emptyset$ and $|A \subset S| = t$, we conclude that $A \cap D = \emptyset$ and further $|B = (A \cup D) \subset (S \cup D)| = n$. Thus the algorithm can recover the polynomial p with the users B. Then we have,

$$\sum_{U_i \in B} \Delta_{i0}^B PK_i = p(0)P = Q.$$

Finally,

$$K = e(H(W), Q)^s = C_2.$$

4.2 Security Proof

Theorem 2. *The proposed threshold broadcast encryption with keyword search scheme is IND-T-CKA secure. If an adversary $\mathcal{A}$ can win Game 1 with the advantage ε, an algorithm $\mathcal{S}$ can be constructed to solve DBDH problem in polynomial time with the advantage $\varepsilon' \geq \frac{\varepsilon}{2e^2(q_C+1)(q_T+1)}$, querying $\mathcal{O}_{Corrupt}$ for at most q_C times and $\mathcal{O}_{Trapdoor}$ for at most q_T times.*

Proof. Let $\delta = (P, aP, bP, cP, T)$ be an instance of DBDH problem (recall definition 1) that a simulator $\mathcal{S}$ is challenged to distinguish that $\delta \in \mathcal{D}_{BDH}$ or $\delta \in \mathcal{D}_{rand}$. From the DBDH instance $\mathbb{D}$, the simulator $\mathcal{S}$ is also given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same order q, a generator P of $\mathbb{G}_1$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. The simulator $\mathcal{S}$ further chooses two hash functions $H : \{0,1\}^* \to \mathbb{G}_1$ and $H' : G_1 \to \mathbb{Z}_q$. Then the simulator $\mathcal{S}$ packs those parameters as $params = (q, \mathbb{G}_1, \mathbb{G}_2.P, e, H, H')$ and passes $param$ to the adversary $\mathcal{A}$.

At the same time, the simulator $\mathcal{S}$ simulates the three oracles as follows.

- $\mathcal{O}_H$: The hash function H is viewed as a random oracle for the adversary $\mathcal{A}$ simulated by the simulator $\mathcal{S}$. Upon requesting the hash value of the keyword W_i, the simulator $\mathcal{S}$ randomly tosses a coin $c_i \in \{0,1\}$ such that $\Pr[c_i = 0] = \alpha$ where α is determined later. The simulator $\mathcal{S}$ also chooses a random value $a_i \in_R \mathbb{Z}_q^+$. Then the simulator $\mathcal{S}$ computes the hash value h_i as
$$h_i = \begin{cases} a_i aP & \text{if } c_i = 0, \\ a_i P & \text{if } c_i = 1. \end{cases}$$
The distribution of $\{h_i\}$ is indistinguishable with a random distribution of $\mathbb{G}_1$. After that, the simulator $\mathcal{S}$ returns h_i to the adversary $\mathcal{A}$. In addition, the simulator $\mathcal{S}$ maintains a hash list $\mathcal{H} = \{W_i, c_i, a_i.h_i\}$. If the requested keyword W is on the list $\mathcal{H}$, the simulator $\mathcal{S}$ returns h_i directly.
- $\mathcal{O}_{KeyGen}$: To create a user U_i, the simulator $\mathcal{S}$ randomly tosses a coin $d_i \in \{0,1\}$ such that $\Pr[d_i = 0] = \beta$ where β is determined later. The simulator also chooses a random value $x_i \in \mathcal{Z}_q^+$. Then the simulator $\mathcal{S}$ computes the secret key SK_i and the public key PK_i as follows.
$$SK_i = \begin{cases} \text{unknown} & \text{if } d_i = 0, \\ x_i & \text{if } d_i = 1. \end{cases} \qquad PK_i = \begin{cases} x_i bP & \text{if } d_i = 0, \\ x_i P & \text{if } d_i = 1. \end{cases}$$
In the case of $d_i = 0$, the secret key $SK_i = x_i b$ cannot be computed by and is unknown to the simulator $\mathcal{S}$ since it is computational hard to compute b from bP. The distribution of $\{PK_i\}$ is indistinguishable with a random distribution of $\mathbb{G}_1$. After that, the simulator $\mathcal{S}$ returns PK_i to the adversary $\mathcal{A}$. In addition, the simulator $\mathcal{S}$ maintains a user key list $\mathcal{K} = \{U_i, d_i, SK_i, PK_i\}$.
- $\mathcal{O}_{Corrupt}$: Upon requesting the secret key SK_i of a created user U_i, the simulator $\mathcal{S}$ searches the user key list $\mathcal{K}$ and checks the corresponding d_i value. If $d_i = 0$, the simulator $\mathcal{S}$ **aborts** since the secret key SK_i is unknown to $\mathcal{S}$. Otherwise, the simulator $\mathcal{S}$ returns SK_i to the adversary $\mathcal{A}$.
- $\mathcal{O}_{Trapdoor}$: To create a created user U_i's trapdoor of a keyword W_j, the simulator $\mathcal{S}$ first looks up the hash list $\mathcal{H}$ for W_j. If $c_j = 0$, the simulator $\mathcal{S}$ simply **aborts**. Otherwise, the simulator $\mathcal{S}$ computes the trapdoor $T = a_j PK_i$ and returns it to the adversary $\mathcal{A}$. The correctness is verified as follows.
$$T = a_j PK_i = \begin{cases} a_j x_i bP = x_i b a_j P = x_i b h_i = SK_i H(W_j) & \text{if } d_i = 0, \\ a_j x_i P = x_i a_j P = x_i h_i = SK_i H(W_j) & \text{if } d_i = 1. \end{cases}$$

Although the trapdoor is still able to be simulated in the case of $c_j = 1 \wedge d_i = 0$ as $T = SK_i H(W_j) = x_i h_j = x_i a_j aP$, the simulator $\mathcal{S}$ still aborts for the simplicity of this proof. In other words, the probability ε' of solving the DBDH problem is greater if the simulator $\mathcal{S}$ does not abort in the above case but makes the proof harder. As long as ε' is not negligible, it is still acceptable.

At some point, the adversary $\mathcal{A}$ outputs a target user set S, a target threshold value t and two target keyword W_0 and W_1. The simulator looks up the hash list $\mathcal{H}$ for W_0 and W_1. If the keyword is not on the list $\mathcal{H}$, the simulator asks the $\mathcal{O}_H$ oracle for its hash value and then the keyword is on the list. If the corresponding values c_0 and c_1 of the keywords W_0 and W_1 is equals to 1, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ randomly picks $b \in \{0, 1\}$ such that $c_b = 0$. If there is only one $c = 0$, the simulator $\mathcal{S}$ has no choice and the value b is fixed. Then we have $H(W_b) = a_b aP$. Due to the restrictions to the adversary $\mathcal{A}$, at least one user U_σ in S has not been corrupted. The simulator $\mathcal{S}$ looks up the user key list $\mathcal{K}$ for that user. If the corresponding value $d_\sigma = 1$, the simulator $\mathcal{S}$ **aborts**. Otherwise, the simulator $\mathcal{S}$ divides the user set S into two sets $S_0 = \{U_i \in S \mid d_i = 0\}$ and $S_1 = \{U_i \in S \mid d_i = 1\}$. Intuitively, $S_0 \neq \emptyset$ because of the existence of U_σ. After that, the simulator $\mathcal{S}$ sets $C_1 = cP$. Before simulating C_2, we first seek how the genuine ciphertext is computed:

$$\begin{aligned}
C_2 &= e(H(W), Q)^c = e(H(W), \sum_{U_i \in S} \Delta_{i0}^S PK_i)^c \\
&= e(H(W), \sum_{U_i \in S_0} \Delta_{i0}^S PK_i)^c \cdot e(H(W), \sum_{U_i \in S_1} \Delta_{i0}^S PK_i)^c \\
&= e(a_b aP, \sum_{U_i \in S_0} \Delta_{i0}^S x_i bP)^c \cdot e(a_b aP, \sum_{U_i \in S_1} \Delta_{i0}^S x_i P)^c \\
&= e(P, P)^{abc \cdot a_b \sum_{U_i \in S_0} \Delta_{i0}^S x_i} \cdot e(aP, cP)^{a_b \sum_{U_i \in S_1} \Delta_{i0}^S x_i}.
\end{aligned}$$

The simulator $\mathcal{S}$ replaces the $e(P, P)^{abc}$ part with T and sets C_2 as

$$C_2 = T^{a_b \sum_{U_i \in S_0} \Delta_{i0}^S x_i} \cdot e(aP, cP)^{a_b \sum_{U_i \in S_1} \Delta_{i0}^S x_i}.$$

After that, the simulator $\mathcal{S}$ selects a set D of $n - t$ dummy users with the same restrictions in the normal construction (i.e. $\mathcal{U} \cap \mathcal{D} = \emptyset$). Similar to C_2, the simulator $\mathcal{S}$ computes K_j as

$$K_j = T^{a_b \sum_{U_i \in S_0} \Delta_{ij}^S x_i} \cdot e(aP, cP)^{a_b \sum_{U_i \in S_1} \Delta_{ij}^S x_i}.$$

Finally, the simulator $\mathcal{S}$ packs the ciphertext $C = (S, t, D, C_1, C_2, \{K_j\}_{U_j \in D})$ and sends to the adversary $\mathcal{A}$. Note that the resulted ciphertext C is consistent only if $T = e(P, P)^{abc}$.

Eventually, the adversary $\mathcal{A}$ outputs a guess b'. If $b = b'$, it means the ciphertext is consistent and it is believed that $T = e(P, P)^{abc}$. Hence, the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{BDH}$. Otherwise, the simulator $\mathcal{S}$ outputs $\delta \in \mathcal{D}_{rand}$.

Lemma 1. *Let ρ be the probability of the simulator $\mathcal{S}$ not aborting. The advantage ε' of the simulator $\mathcal{S}$ solving the DBDH problem is at least $\frac{\rho\varepsilon}{2}$, assuming the probability of $\delta \in \mathcal{D}_{BDH}$ is $\frac{1}{2}$ and the adversary $\mathcal{A}$ wins Game 1 with the advantage ε.*

Proof. We prove this lemma by calculating the probability of the simulator $\mathcal{S}$ succeeding. If $\delta \in \mathcal{D}_{rand}$, the behaviour of the adversary $\mathcal{A}$ is unpredictable. Thus, the simulator $\mathcal{A}$ succeeds at least better than a random guess with succeeding probability of $\frac{1}{2}$. Similarly, if the simulator S aborts, we just have a random guess. Otherwise, we take the result of the adversary $\mathcal{A}$ with the correct probability of $\frac{1}{2} + \varepsilon$. Let R be the event that the simulator $\mathcal{S}$ succeeds with a random guess. We have,

$$
\begin{aligned}
&\Pr[\mathcal{S} \text{ succeeds}] \\
&= \frac{1}{2}\Pr[\mathcal{S} \text{ succeeds} \mid \delta \in \mathcal{D}_{BDH}] + \frac{1}{2}\Pr[\mathcal{S} \text{ succeeds} \mid \delta \in \mathcal{D}_{rand}] \\
&\geq \frac{1}{2}\left(\Pr[\mathcal{S} \text{ does not abort}] \cdot \Pr[\mathcal{A} \text{ succeeds}] + \Pr[\mathcal{S} \text{ aborts}] \cdot \Pr[R]\right) + \frac{1}{2}\Pr[R] \\
&= \frac{1}{2}\left(\Pr[\mathcal{S} \text{ does not abort}] \cdot (\frac{1}{2} + \varepsilon) + \Pr[\mathcal{S} \text{ aborts}] \cdot \frac{1}{2}\right) + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{2}\left(\rho \cdot (\frac{1}{2} + \varepsilon) + (1-\rho) \cdot \frac{1}{2}\right) + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{2}\rho\varepsilon + \frac{1}{2}.
\end{aligned}
$$

Since $\Pr[\mathcal{S} \text{ succeeds}] \geq \frac{1}{2}\rho\varepsilon + \frac{1}{2}$ and $Adv_{\mathcal{S}}^{DBDH} = \left|\mathcal{S} \text{ succeeds}] - \frac{1}{2}\right|$, we have

$$Adv_{\mathcal{S}}^{DBDH} \geq \frac{\rho\varepsilon}{2}.$$

Lemma 2. *The simulator $\mathcal{S}$ does not abort with the probability ρ at least*

$$\frac{1}{e^2(q_C+1)(q_T+1)},$$

querying $\mathcal{O}_{Corrupt}$ for at most q_C times and $\mathcal{O}_{Trapdoor}$ for at most q_T times.

Proof. There are 4 possible points that the simulator $\mathcal{S}$ may abort.

1. The simulator $\mathcal{S}$ aborts in answering $\mathcal{O}_{Corrupt}$ queries if $d_i = 0$. The single abort probability is β. The probability of not aborting for all $\mathcal{O}_{Corrupt}$ queries is $\Pr[E_1] = (1-\beta)^{q_C}$.
2. The simulator $\mathcal{S}$ aborts in answering $\mathcal{O}_{Trapdoor}$ queries if $c_i = 0$. The single abort probability is α. The probability of not aborting for all $\mathcal{O}_{Trapdoor}$ queries is $\Pr[E_2] = (1-\alpha)^{q_T}$.
3. The simulator $\mathcal{S}$ aborts in the challenge phase if $c_0 = c_1 = 1$. The probability of not aborting for this event is $\Pr[E_3] = 1 - (1-\alpha)^2 = 2\alpha - \alpha^2$. Since $\alpha \in [0,1]$, we have

$$\alpha \leq 1 \implies \alpha^2 \leq \alpha \implies 0 \leq \alpha - \alpha^2 \implies \alpha \leq 2\alpha - \alpha^2 = \Pr[E_3].$$

4. The simulator $\mathcal{S}$ aborts in the challenge phase if $d_\delta = 1$. The probability of not aborting for this event is $\Pr[E_4] = \beta$.

Since all the events are independent, we have

$$\rho = \Pr[E_1] \cdot \Pr[E_2] \cdot \Pr[E_3] \cdot \Pr[E_4] \geq \alpha(1-\alpha)^{q_T}\beta(1-\beta)^{q_C}.$$

The function $\alpha(1-\alpha)^{q_T}\beta(1-\beta)^{q_C}$ is maximised when $\alpha = \frac{1}{q_T+1}$ and $\beta = \frac{1}{q_C+1}$. Now we have

$$\begin{aligned}
\rho &\geq \frac{1}{q_T+1}(1-\frac{1}{q_T+1})^{q_T} \cdot \frac{1}{q_C+1}(1-\frac{1}{q_C+1})^{q_C} \\
&\geq \frac{1}{q_T+1}\left(\lim_{q_T\to\infty}(1-\frac{1}{q_T+1})^{q_T}\right) \cdot \frac{1}{q_C+1}\left(\lim_{q_C\to\infty}(1-\frac{1}{q_C+1})^{q_C}\right) \\
&= \frac{1}{e^2(q_C+1)(q_T+1)}.
\end{aligned}$$

Note that the statement $(1-\frac{1}{q_T+1})^{q_T} \geq \frac{1}{e} = \lim_{q_T\to\infty}(1-\frac{1}{q_T+1})^{q_T}$ is always true for any q_T. Similar statement for q_C also applies.

Combining $Adv_{\mathcal{S}}^{DBDH} \geq \frac{\rho\varepsilon}{2}$ from lemma 1 and $\rho \geq \frac{1}{e^2(q_C+1)(q_T+1)}$ from lemma 2, we have

$$\varepsilon' = Adv_{\mathcal{S}}^{DBDH} \geq \frac{\varepsilon}{2e^2(q_C+1)(q_T+1)}.$$

5 Conclusion

In this paper, we defined Threshold Broadcast Encryption with Keyword Search (TBEKS) scheme and its IND-T-CKA security model. We proposed the first TBEKS scheme and proved it is IND-T-CKA secure, assuming the hardness of the Decisional Bilinear Diffie-Hellman problem. In our TBEKS scheme, we consider the server to be honest but curious and we do not allow the server to collude with the users. It is an open problem to build a scheme that is secure against a malicious server that may collude with other users.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. Journal of Cryptology 21(3), 350–391 (2008)
2. Attrapadung, N., Furukawa, J., Imai, H.: Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In: Lai, X., Chen, K. (eds.) Advances in Cryptology – ASIACRYPT 2006, Lecture Notes in Computer Science, vol. 4284, pp. 161–177. Springer Berlin Heidelberg (2006)
3. Baek, J., Safavi-Naini, R., Susilo, W.: On the integration of public key data encryption and public key encryption with keyword search. In: Katsikas, S., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) Information Security, Lecture Notes in Computer Science, vol. 4176, pp. 217–232. Springer Berlin Heidelberg (2006)

4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology - EUROCRYPT 2004, Lecture Notes in Computer Science, vol. 3027, pp. 506–522. Springer Berlin Heidelberg (2004)
5. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer Berlin Heidelberg (2001)
6. Daza, V., Herranz, J., Morillo, P., Ràfols, C.: Cca2-secure threshold broadcast encryption with shorter ciphertexts. In: Susilo, W., Liu, J., Mu, Y. (eds.) Provable Security, Lecture Notes in Computer Science, vol. 4784, pp. 35–50. Springer Berlin Heidelberg (2007)
7. Hwang, Y., Lee, P.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing-Based Cryptography – Pairing 2007, Lecture Notes in Computer Science, vol. 4575, pp. 2–22. Springer Berlin Heidelberg (2007)
8. Mell, P., Grance, T.: The nist definition of cloud computiing. Tech. rep., National Institue of Standards and Technology (2011)
9. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (Nov 1979)
10. Siad, A.: Anonymous identity-based encryption with distributed private-key generator and searchable encryption. In: New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on. pp. 1–8 (May 2012)
11. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on. pp. 44–55 (2000)
12. Sun, W., Yu, S., Lou, W., Hou, T., Li, H.: Protecting your right: Verifiable attribute-based keyword search with fine-grainedowner-enforced search authorization in the cloud. Parallel and Distributed Systems, IEEE Transactions on PP(99), 1–1 (2014)
13. Wang, P., Wang, H., Pieprzyk, J.: Threshold privacy preserving keyword searches. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008: Theory and Practice of Computer Science, Lecture Notes in Computer Science, vol. 4910, pp. 646–658. Springer Berlin Heidelberg (2008)
14. Xu, P., Jin, H., Wu, Q., Wang, W.: Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. Computers, IEEE Transactions on 62(11), 2266–2277 (Nov 2013)
15. Zheng, Q., Xu, S., Ateniese, G.: Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. In: INFOCOM, 2014 Proceedings IEEE. pp. 522–530 (April 2014)