# Certificate-Based Encryption with Keyword Search

Enabling Secure Authorization in Electronic Health Record

Clémentine Gritti, Willy Susilo*, and Thomas Plantard
Centre for Computer and Information Security Research
School of Computing and Information Technology
University of Wollongong, Australia
cjpg967@uowmail.edu.au, {wsusilo, thomaspl}@uow.edu.au

**Abstract**

In an e-Health scenario, we study how the practitioners are authorized when they are requesting access to medical documents containing sensitive information. Consider the following scenario. A clinician wants to access and retrieve a patient's Electronic Health Record (EHR), and this means that the clinician must acquire sufficient access right to access this document. As the EHR is within a collection of many other patients, the clinician would need to specify some requirements (such as a keyword) which match the patient's record, as well as having a valid access right. The complication begins when we do not want the server to learn anything from this query (as the server might be outsourced to other place). To encompass this situation, we define a new cryptographic primitive called Certificate-Based Encryption with Keyword Search (CBEKS), which will be suitable in this scenario. We also specify the corresponding security models, namely computational consistency, indistinguishability against chosen keyword and ciphertext attacks, indistinguishability against keyword-guessing attacks and collusion resistance. We provide a CBEKS construction that is proven secure in the standard model with respect to the aforementioned security models.

**Keywords**: Public-Key Encryption with Keyword Search, Certificate-Based Encryption, Consistency, Indistinguishability, Collusion Resistance.

## 1 Introduction and Motivation

Authorizing medical staff members to access and retrieve medical documents brings security and privacy issues. We present three scenarios that describe the authorization process in a hospital to enable medical staff members to acquire access to patients' Electronic Health Records (EHRs).

Let's consider our first scenario as follows. A patient, Betty, visits the hospital for a gynecological checkup. She requests that the checkup will be conducted by a female clinician. On her arrival, a nurse, Alice, takes Betty's blood sample and records this information "securely" in Alice's EHR, which is stored in the local server of the hospital. The security of the EHR is done by encrypting this information prior to uploading it to the local server. Once this is done, Alice will need to wait until she is called by the available female gyneacologist as requested.

Let Carol be a female gyneacologist in the hospital. She has to access Betty's EHR, and in particular the blood sample report, in order to discuss with the patient on some potential issues. To do so, Carol sends a request to the server that contains a descriptive keyword for the record that she needs to access (e.g. "Betty" in this example) and some information about her access right (e.g. she is a gynecologist in the hospital). The information will only be delivered (or decryptable) by Carol if her access right is valid

*Journal of Internet Services and Information Security (JISIS)*, volume: 6, number: 4 (November 2016), pp. 1-34

*Corresponding author: Northfields Avenue Wollongong NSW 2522, Australia, Tel: +61-2-4221-5535, Web: http://www.uow.edu.au/~wsusilo/

and the data that she is looking for (i.e. Betty's EHR) is available on the local server. This scenario is depicted in Fig. 1.

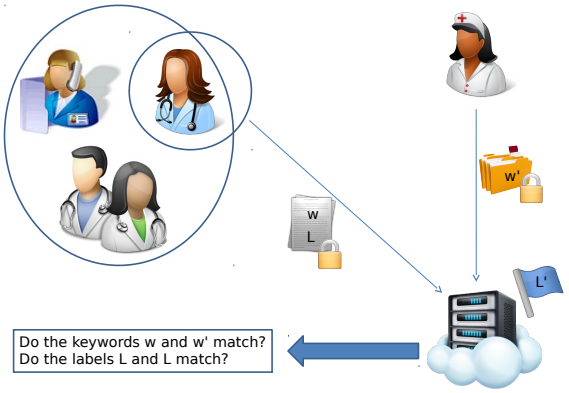


Figure 1: Alice uploads Betty's records on the server of the hospital. The records contain an encryption of the keyword $w'$ that she has chosen to describe them. Carol sends a file encrypting her keyword $w$ and her label $L$ (corresponding to her access right) as a request to retrieve Betty's records. The server holds a label $L'$ according to the hospital regulation. If the keywords and the labels match, then the server released the requested records to Carol.

Our second scenario takes place in pediatrics and kids hospital. A child, Dan, has contracted chickenpox and his parents decide to bring him to the hospital to check his condition. Since this disease is also contagious to adults and other children, Dan has to been examined by a pediatrician who is immune against chickenpox. Thus, this requirement has to be added to the clinician's access right to let this clinician be able to retrieve Dan's EHR. At the end of the diagnosis, the clinician will also need to store the result of the examination in Dan's EHR.

Our last scenario is due to the communciation difficulty. Eva, who is a Russian tourist and currently visiting her 80-year-old uncle in Australia, visits the emergency department due to her broken ankle. Unfortunately, Eva does not speak English but Russian, and she requires a clinician that speaks her language. Fortunately, an orthopedist, Frank, satisfies this requirement. By providing his access right that includes this linguistic feature and a descriptive keyword, Frank will be able to retrieve Eva's EHR.

These scenarios illustrate a required framework to authorize medical staff members to access and retrieve the necessary medical documents to conduct their job. As illustrated earlier, there are two essential information pieces required, namely the *access right* of a clinician that has to be in accordance with the hospital requirements (regarding the patients' specific requirements for instance) and a *keyword* selected by the clinician as the description of the requested medical documents.

### 1.1 Basic Technical Settings

In order to capture the above scenarios, we consider four entities:

1. An *uploader* that transfers medical data contents (e.g. EHRs) to a server.

2. A *server* that stores the uploader's encrypted medical data contents and delivers these contents to the receivers if and only if certain conditions are met.

3. Several *receivers*, that ask the server to retrieve medical data contents.

4. A *certifier* that delivers certificates to the receivers according to their access right status.

In practice, the receivers are the medical staff of a hospital while the uploader can be anyone, the certifier is the IT department of the hospital and the server is the hospital database. The assumption is that we do not want the administrator of the server to be able to read the documents without proper consents, and therefore the data will be encrypted. Hence, we adopt the honest-but-curious model for the server.

We suppose that Alice, a nurse working at the hospital, needs to access some medical documents. To do so, Alice chooses a keyword $w^R$ describing the documents that she wants to retrieve, creates a trapdoor that is a function of the keyword $w^R$ and a certificate, and sends this trapdoor to the server as an access request.

Alice is equipped with a valid certificate in order to conduct this operation. Alice's certificate is updated regularly from the broadcasted information provided by the certifier. The update key is created given a group $S$ of medical staff members and a label $L^R$. Note that the update key is broadcasted to all the people working at the hospital but only people belonging to the group $S$ can successfully update their certificates. In addition, the label $L^R$ refers to information such as access rights and other privileges and features, and enables the staff members in $S$ to access and retrieve some documents. Note that the certificate needs to be refreshed since the label $L^R$ and/or the group $S$ might change over the time.

When Alice sends the trapdoor to the server, the latter checks it with a ciphertext encrypting a keyword $w^S$ that describes the requested documents (the keyword $w^S$ was chosen by the person who encrypted the documents) and the current label $L^S$ (the label $L^S$ is determined by the hospital). The server is actually verifying whether the keywords and the labels match (i.e. whether $w^R = w^S$ and $L^R = L^S$). If so, then the server releases the encrypted documents to Alice; otherwise, the server keeps them on its local storage.

We illustrate our CBEKS protocol in Fig. 2, 3 and 4 such that the four entities are depicted. First, the certifier generates the first certificate given to each receiver as an encryption of a label $L_1^R$ (Fig. 2). It also computes update keys that are sent to all the receivers. Receivers can successfully refresh their certificates using the update keys if and only if they belong to a group $S_j$ specified by the certifier (Fig. 3). Then, the uploader computes a ciphertext $\mathscr{CT}_{w^S}$ for a keyword $w^S$ and transmits it to the server. Each receiver computes a trapdoor $Trap_{w_i^R, L_j^R}$ according to a keyword $w_i^R$ and the most recent certificate encrypting the label $L_j^R$; the receiver then forwards it to the server. At last, the server checks whether the two keywords match and whether the label embedded into the trapdoor is equal to the current label $L_l^S$, i.e. $w_i^R = w^S$ and $L_j^R = L_l^S$ (Fig. 4).

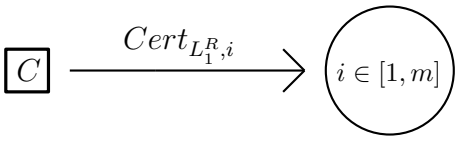


Figure 2: A certifier $C$ generates a first certificate $Cert_{L_1^R,i}$ for a label $L_1^R$, and gives it to a receiver $i$, where $i \in [1,m]$.

## 1.2 Comparisons with Existing Cryptographic Primitives

Attribute-Based Encryption (ABE) [17] is a cryptographic primitive that involves attributes to generate secret keys and ciphertexts. Attributes can be seen as the components describing the access right of a receiver. Observe that no certificate is delivered and no key update is possible in an ABE system. Instead, secret keys are created with respect to each attribute taken individually and have to be re-generated from scratch each time that an attribute is added, deleted or modified. We note that treating the attributes individually each time is cumbersome and inefficient. Moreover, an ABE system is not equipped with searching capability: no keyword or other requirement is necessary to complete the authorization step. Thus, such primitive does not seem suitable to satisfy the medical criteria cited above.

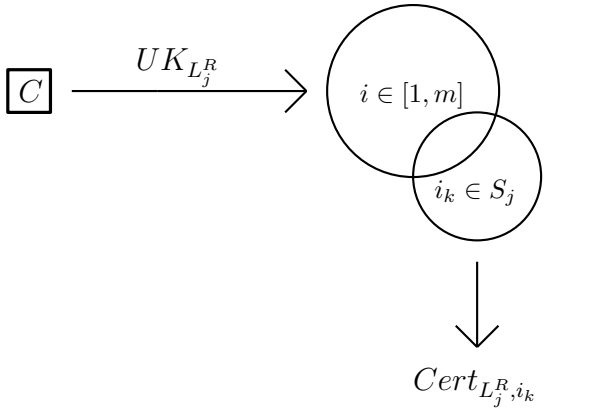


Figure 3: The certifier $C$ computes an update key $UK_{L_j^R}$ for $L_j^R$ and a group $S_j \subseteq [1,m]$ of receivers, and gives it to all the receivers $i \in [1,m]$. Then, a receiver $i$ can successfully use $UK_{L_j^R}$ to obtain a refreshed certificate $Cert_{L_j^R,i}$ if and only if $i \in S_j$.

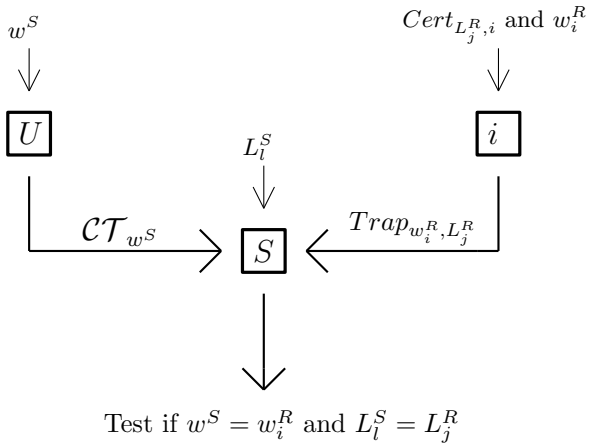


Figure 4: Afterwards, the uploader $U$ generates a ciphertext $\mathscr{CT}_{w^S}$ for a keyword $w^S$, and transfers it to a server $S$. With inputs $w_i^R$ and $Cert_{L_j^R,i}$ for $L_j^R$, the receiver $i$ computes a trapdoor $Trap_{w_i^R,L_j^R}$ and forwards it to the server $S$, where $i \in [1,m]$. Finally, given $L_l^S$, $\mathscr{CT}_{w^S}$ and $Trap_{w_i^R,L_j^R}$, the server $S$ tests whether $w^S = w_i^R$ and $L_l^S = L_j^R$.

In our previous work [18], we proposed a Certificate-Based Broadcast Encryption (CBBE) protocol, called File Sharing in Electronic Health Records (FSEHR). This protocol enables medical staff members (receivers) to communicate among themselves, as well as to retrieve medical documents. Authorization is given through licences (certificates) that allow the medical staff to practice in the given hospital. We observe that a FESHR system does not satisfy the medical scenario described in this paper. First, certificates cannot be collectively updated, but have to be individually re-generated in case of access right changes. Moreover, a receiver does not search for medical documents, but acquires access to all of them after successful authorization (even the ones that this receiver is not interested to acquire).

### 1.3 Our Work

In this paper, we address the problem of authorizing receivers in a sensitive environment to let them access and retrieve private documents securely. In order to access data encrypted by an uploader and stored on a server, a receiver has to provide a trapdoor that embeds two elements: a keyword that describes the targeted data and a certificate that includes a label as the access right of the receiver. The receiver will retrieve the encrypted data if and only if the keyword that it has chosen is the same than the one defined by the uploader and if the label specifying the access right of the receiver is the same than the current one held by the server. To do so, we define a new cryptographic primitive that we call Certificate-Based Encryption with Keyword Search (CBEKS). We also specify the corresponding security models, namely computational consistency, indistinguishability against chosen keyword and ciphertext attacks, indistinguishability against keyword-guessing attacks and collusion resistance. We provide a CBEKS construction that is proven secure in the standard model with respect the aforementioned security mod-

els.

At a glance, by just simply observing the name of the primitive *Certificate-Based Encryption with Keyword Search*, one may think that this is a trivial combination between a Certificate-Based Encryption (CBE) scheme and a Public-Key Encryption with Keyword Search (PEKS). Unfortunately, this is not the case. This is due to the fact that a CBE scheme [15] requires an interaction between one single certifier and one single receiver. This is not suitable for CBEKS as we involve many receivers. Furthermore, the certificates cannot be refreshed using update keys, but rather they need to be re-generated every time. Hence, the direct use of CBE scheme will not be satisfactory. More importantly, the CBE scheme in [15] only works in the random oracle model, and to make our CBEKS scheme useful in practice, the scheme must be proven secure in the standard model (i.e. to guarantee the security since CBEKS deals with sensitive data). Another attempt is to combine Fang et al.'s secure-channel free PEKS scheme [12] and our work in [18], which provides the broadcast mechanism. Again, the resulting scheme will not suffice to satisfy the requirements in CBEKS with a simple modification, and more importantly, if the modification is successul then the scheme will again be only secure in the random oracle model.

We summarize the advantages and disadvantages of the aforementioned primitives and combinations of primitives in Table 1. Observe that our CBEKS system satisfies all the requirements and features highlighted previously.

| Cryptographic Primitive | Certificate | Broadcast | Key update | Keyword |
|---|---|---|---|---|
| ABE [17] | × | × | × | × |
| CBBE [18] | ✓ | ✓ | × | × |
| PEKS [12] + CBE [15] | ✓ | × | × | ✓ |
| PEKS [12] + CBBE [18] | ✓ | ✓ | × | ✓ |
| CBEKS | ✓ | ✓ | ✓ | ✓ |

Table 1: Comparisons of the new primitive CBEKS, the existing primitives ABE and CBBE and the combination of the primitive PEKS with the primitives CBE and CBBE respectively.

### 1.4 Related Work

Boneh et al. [5] introduced the notion of Public-Key Encryption with Keyword Search (PEKS) in order to solve the problem of searching on encrypted data in a public key environment. Then, Baek et al. [2] obtained the new primitive Public-Key Encryption (PKE)/PEKS by combining a PEKS scheme [5] and an ElGamal encryption scheme [10]. Thereafter, several works followed on PEKS and related primitives such as PEKS with a designated tester (dPEKS) [7, 14, 19, 11, 27, 28, 26, 31]. Baek et al. [3] improved the concept of PEKS by defining the notion of Secure-Channel-Free PEKS (SCF-PEKS). Recently, Fang et al. [12] gave a SCF-PEKS scheme that is secure against chosen keyword and ciphertext attacks and keyword-guessing attacks without random oracle, meaning that this scheme achieves the strongest security levels.

Fiat and Naor [13] introduced the concept of BE. In such a scheme, a broadcaster wants to share a message with a dynamically chosen group $S$ of receivers. To do that, it generates a ciphertext such that only authorized receivers in $S$ can decrypt it. Then, Naor et al. [22] presented a symmetric-key system and gave both security model and security analysis. Dodis and Fazio [9] designed the first public-key BE scheme secure against ciphertext-chosen attacks. Later, Boneh et al. [6] proposed a fully collusion resistant scheme proved selectively secure in the standard model. They obtained a scheme with short secret

keys and ciphertexts by applying computational techniques using groups with bilinear maps. Thereafter, several works followed [8, 30, 16, 24, 25].

Naor and Nissim [23] proposed a solution to overcome the problem of certificate revocation. Using an authenticated search data structure, they constructed certificate revocation lists in order to store, update and retrieve authenticated information related to certificates. Then, Gentry [15] introduced the concept of CBE such that a certificate can be seen as a decryption key. More precisely, a receiver should use both its secret key and its certificate to decrypt a ciphertext. The certificate is delivered by a Certificate Authority (CA) and embeds time periods that tell when the receiver is allowed to retrieve messages.

## 2 Protocol Definition

In the definition of the protocol, a label is a reference to some information about access rights (e.g. privileges and features). This label is supposed to be a unique element in $\mathbb{Z}_p$, for a prime number $p$, meaning that it refers to only one collection of rights and two labels with different collections cannot be equal.

We assume that the certifier and the server know the labels; however, none of the receivers should get any information about these elements. We presume that the certifier and the server communicate securely to agree on all the possible labels. We do not consider how they communicate in this protocol by making the hypothesis that they can proceed easily and naturally. For instance, in a hospital, let the certifier be the IT department and the server be the database center, meaning that they both belong to the administration section of the organization. Thus, there exists a way for the certifier and the server to share the information about the labels, since they are allowed to know the access rights and the privileges given to the medical staff members.

A Certificate-Based Encryption with Keyword Search (CBEKS) protocol comprises the following algorithms:

$\mathsf{Setup}(\lambda, m) \rightarrow (params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]})$. This probabilistic algorithm is run by a trusted group manager to setup the protocol. On inputs a security parameter $\lambda$ and a total number of receivers $m$, output the public parameters *params*, the server's secret key $sk_S$, the certifier's secret key $sk_C$ and the receivers' secret keys $\{sk_{R,i}\}_{i\in[1,m]}$.

As suggested above, the server and the certifier might receive information about the label framework. The public parameters *params* include the public keys of all the involved entities.

$\mathsf{Encrypt}(params, w^S) \rightarrow \mathscr{CT}_{w^S}$. This probabilistic algorithm is run by the uploader to generate the ciphertext. On inputs the public parameters *params* and a uploader's keyword $w^S$, output the ciphertext $\mathscr{CT}_{w^S}$ for $w^S$.

Note that the keyword $w^S$ is chosen by the uploader that encrypts the message $\mathscr{M}$ (e.g. a patient's EHR) and uploads the resulting ciphertext $\mathscr{CT}_{\mathscr{M}}$ along with $\mathscr{CT}_{w^S}$ on the server. (In this paper, we do not focus on the encryption process for $\mathscr{CT}_{\mathscr{M}}$ but rather on the encryption process for $\mathscr{CT}_{w^S}$.)

$\mathsf{CertGen}(params, sk_C, L_1^R, i) \rightarrow Cert_{L_1^R,i}$. This probabilistic algorithm is run by the certifier to generate a receiver $i$'s first certificate. On inputs the public parameters *params*, the certifier's secret key $sk_C$, a label $L_1^R$ and a receiver $i$, output the receiver $i$'s first certificate $Cert_{L_1^R,i}$ for $L_1^R$.

The certifier might keep some elements used to generate $Cert_{L_1^R,i}$, such that the label $L_1^R$ and additional information, on its local storage in order to create the update key $UK_{L_2^R}$. Let $AI_{L_1^R}$ be the auxiliary information that the certifier stores.

$\mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j) \rightarrow UK_{L_j^R}$. This probabilistic algorithm is run by the certifier to generate the update key for a group $S_j \subseteq [1,m]$ of receivers. On inputs the public parameters *params*, the certifier's secret key $sk_C$, the auxiliary information $AI_{L_{j-1}^R}$, a label $L_j^R$ where $1 < j$ and a

group $S_j$ of receivers, output the update key $UK_{L_j^R}$ for $L_j^R$ and $S_j$.

The certifier might keep some elements used to generate $UK_{L_j^R}$, such that the label $L_j^R$ and additional information, on its local storage in order to create the update key $UK_{L_{j+1}^R}$. Let $AI_{L_j^R}$ be the auxiliary information that the certifier stores.

The algorithm $\mathsf{UpdtKeyGen}$ is run in three cases:

1. There is a new label $L_j^R$ replacing $L_{j-1}^R$ while $S_j = S_{j-1}$.
2. There is a new group $S_j$ replacing $S_{j-1}$ while $L_j^R = L_{j-1}^R$.
3. There are a new label $L_j^R$ replacing $L_{j-1}^R$ and a new group $S_j$ replacing $S_{j-1}$.

In all cases, we write $AI_{L_{j-1}^R}$, $L_j^R$ and $S_j$ as the inputs for the algorithm $\mathsf{UpdtKeyGen}$, such that $AI_{L_{j-1}^R}$ was computed given $L_{j-1}^R$ and $S_{j-1}$. (For $j = 2$, $AI_{L_1^R}$ corresponds to the auxiliary information from $Cert_{L_1^R,i}$, for $i \in [1,m]$.) We assume that a description of the group $S_j$ cam be found in $UK_{L_j^R}$.

$\mathsf{UpdtCert}(params, Cert_{L_{j-1}^R,i}, UK_{L_j^R}) \rightarrow Cert_{L_j^R,i}$. This deterministic algorithm is run by a receiver $i \in [1,m]$ to refresh its certificate. On inputs the public parameters *params*, the receiver $i$'s previous certificate $Cert_{L_{j-1}^R,i}$ for $L_{j-1}^R$ and the key update $UK_{L_j^R}$ for $L_j^R$ and $S_j$ where $1 < j$, output the receiver $i$'s refreshed certificate $Cert_{L_j^R,i}$ for $L_j^R$ if $i \in S_j$; output $\perp$ otherwise.

$\mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i}) \rightarrow Trap_{w_i^R,L_j^R}$. This probabilistic algorithm is run by a receiver $i \in [1,m]$ to generate the trapdoor. On inputs the public parameters *params*, the receiver $i$'s secret key $sk_{R,i}$, a receiver $i$'s keyword $w_i^R$ and a receiver $i$'s certificate $Cert_{L_j^R,i}$ for $L_j^R$, output the receiver $i$'s trapdoor $Trap_{w_i^R,L_j^R}$ for $w_i^R$ and $L_j^R$.

$\mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R,L_j^R}) \rightarrow b$. This deterministic algorithm is run by the server to check that the keywords and the labels match. On inputs the public parameters *params*, the server's secret $sk_S$, the ciphertext $\mathscr{CT}_{w^S}$ for $w^S$, a label $L_l^S$ where $1 \leq l$ and the receiver $i$'s trapdoor $Trap_{w_i^R,L_j^R}$ for $w_i^R$ and $L_j^R$ where $1 \leq j$, ouptut $b = 1$ if $[w^S = w_i^R] \wedge [L_l^S = L_j^R]$; output $b = 0$ otherwise.

**Correctness** For all $(params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}) \leftarrow \mathsf{Setup}(\lambda, m)$, let the ciphertext be $\mathscr{CT}_{w^S} \leftarrow \mathsf{Encrypt}(params, w^S)$ for a keyword $w^S$. Let a receiver $i \in [1,m]$ have a first certificate $Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i)$ for a label $L_1^R$. Given an update key $UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j)$ such that $i \in S_j \subseteq [1,m]$, and a previous certificate $Cert_{L_{j-1}^R,i}$ for a label $L_{j-1}^R$, the certificate is refreshed as follows: $Cert_{L_j^R,i} \leftarrow \mathsf{UpdtCert}(params, Cert_{L_{j-1}^R,i}, UK_{L_j^R})$ for a label $L_j^R$ where $1 < j$. Then, let the receiver $i$ create a trapdoor $Trap_{w_i^R,L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i})$ for a keyword $w_i^R$ and the label $L_j^R$. If $j = 1$, then we simply use $Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i)$ during the trapoor generation. For a label $L_l^S$, if $[w^S = w_i^R] \wedge [L_l^S = L_j^R]$ where $1 \leq j, l$, then $\mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R,L_j^R})$ outputs 1; otherwise, $\mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R,L_j^R})$ outputs 0.

# 3 Security Models

Before describing the security levels that we expect for our scheme, we recall the existing security models in the literature. In [12], the SCF-PEKS scheme is proved secure against chosen keyword and ciphertext attacks (IND-CKCA) and against keyword-guessing attacks (IND-KGA) in the standard model, as well as is proved computationally consistent. More precisely, the scheme proposed in [12] is proven secure in terms of indistinguishability under chosen keyword and ciphertext attacks, meaning that

1. the server that has not obtained the trapdoors for given keywords cannot tell which ciphertext encrypts which keyword;

2. the receiver that has not obtained the server's secret key cannot make any decisions about the ciphertexts, even though it gets all the trapdoors for the keywords that it holds.

In addition, a security proof is given in [12] to cover the notion of indistinguishability under keyword-guessing attack, that guarantees that an outsider (neither the server nor the receiver) that has obtained the trapdoor for a challenge keyword cannot observe the link between the trapdoor and any keywords. We will demonstrate that our CBEKS scheme reaches similar security notions regarding keywords and ciphertexts that we adapt to deal with labels and certificates.

In [6], a Broadcast Encryption (BE) scheme achieves fully collusion resistance (CR). More precisely, this system can broadcast a session key to any group of receivers and remains secure even if malicious unauthorized receivers collude. Such a property should be achieved to deal with the fact that the certifier forwards an update key to all the receivers, such that only a group of authorized receivers will successfully refresh their certificates.

We now give an overview of the threats and attacks that our CBEKS system should elude:

**Security against the Server:** The server handles the labels for the test process, meaning that it can obtain information from the certificates about the access rights given to the receivers. However, the server should learn nothing about the uploader's keyword (through the ciphertext) and the receivers' keyword (through the trapdoor), except whether they match or not. This is formalized in the IND-CKCA game played by the server.

**Security against the Certifier:** As the server, the certifier knows the labels since it has the task to create the first certificates and update keys. Nevertheless, it should not be able to update itself the receivers' certificates. Moreover, even if intercepting ciphertexts and trapdoors, the certifier should not get any information about the keywords chosen by the uploader and the receivers respectively. This is formalized in the IND-CKCA game played by the certifier.

**Security against the Receiver:** The receiver gets the first certificate from the certifier, along with update keys, such that the latters can be efficiently used if and only if the receiver has been authorized by the certifier. The receiver should not be able to learn anything about the labels and so, the access rights embedded into its first certificate and the subsequent update keys. Note that the receiver can guess whether its refreshed certificate is a correct one or a fake one, since we suppose that the group of authorized receivers is contained in clear into each update key. Even waylaying a ciphertext, a receiver should not have the capability to know the embedded uploader's keyword, and even check that whether the keywords match. This is formalized in the IND-CKCA game played by the receiver.

Moreover, observe that the trapdoor is generated given a keyword and a certificate to avoid the following collusion attack. We suppose that the trapdoor only encrypts a keyword and that a receiver has to provide both its trapdoor and its certificate to the server in order to verifiy the matches of the keywords and the labels. In this scenario, we can let a first receiver compute the trapdoor encrypting a keyword and have only obsolete certificates, and a second receiver get a recent certificate that is still valid. Thus, these two receivers can manage to pass the test by sending to the server the trapdoor and the fresh certificate respectively.

**Security against an Outsider:** An outsider is neither the server nor the certifier nor a receiver. This outsider will guess keywords (for instance, keywords with low entropy) and check its choices in an off-line way. If the outsider has successfully initiated a keyword-guessing attack, then it can learn which keywords were chosen by the uploader and by the receivers, and so the security of the protocol might be broken. This is formalized in the IND-KGA game.

**Collusion Resistance:** The update keys are delivered by the certifier in order to let a group of authorized receivers to resfresh their certificates, regarding either a new label or a new group or both. This group is supposed to be included into the update key in clear, and the latter is sent to all the receivers. One important feature that the update key should satisfy is its collusion resistance: even if all the unauthorized receivers collude, they cannot generate a well-formed refreshed certificate from the update key. This is formalized in the CR game.

We provide several security games where the adversary plays the role of either the server or the certifier or a receiver. We also give a security game when the adversary acts as an outsider (neither the server nor the certifier nor a receiver) or as a group of colluding receivers.

The security models that we define below are computational consistency, indistinguishability against chosen keyword and ciphertext attack (IND-CKCA), indistinguishability against keyword-guessing attack (IND-KGA) and collusion resistance (CR). Compared to the IND-CKCA and IND-KGA models given in [12], the adversary has access to more oracles in our case, in order to satisfy the label-based situation of our protocol. Informally, in addition to the trapdoor queries and the test queries, the adversary can make first certificate queries, update key queries and refreshed certificate queries. If the adversary makes an update key query or a refreshed certificate query for a label $L_j^R$, then the challenger computes the requested element using the previous queried label $L_{j-1}^R$ or a random label $L_{j-1}^R$ for a first query.

In the collusion resistance game, we let the adversary select a group $S^* \subseteq [1,m]$ of receivers beforehand, and the challenger will reply to the adversary's queries according to this group $S^*$.

In summary, depending on the role that it is playing, the adversary is given access to different oracles:

*First certificate query:* the adversary can ask the challenger for a first certificate query by giving a label $L$. The challenger responds by sending back a first certificate $Cert$ for $L$ to the adversary.

*Update key query:* the adversary can ask the challenger for an update key query by giving a label $L$. The challenger chooses a group of receivers $S$ and responds by sending back an update key $UK$ for $L$ and $S$ to the adversary.

*Refreshed certificate query:* the adversary can ask the challenger for a refreshed certificate query by giving a label $L$. The challenger responds by sending back a refreshed certificate $Cert$ for $L$ to the adversary.

*Trapdoor query:* the adversary can ask the challenger for a trapdoor query by giving a keyword $w$ and a label $L$. The challenger responds by sending back a trapdoor $Trap$ for $w$ and $L$ to the adversary.

*Test query:* the adversary can ask the challenger for a test query by giving a ciphertext $\mathscr{CT}$, a keyword $w$ and two labels $L, L'$. The challenger responds by sending back the result $b \in \{0,1\}$ to the adversary.

## 3.1 Consistency

The definition of consistency follows the ones given in [12, 1] except that the adversary has to choose more elements along with the two keywords: along with the uploader's keyword $w^S$, it selects a corresponding label $L_l^S$ where $1 \leq l$, and along with the receiver's keyword $w_i^R$, it first selects a receiver $i \in [1,m]$ and then a label $L_1^R$ as well as an index $1 < j$ indicating the number of times that the certificate should be refreshed.

Let $\lambda$ be the security parameter and $m$ the total number of receivers. Suppose there exists an adversary $\mathscr{A}$ that wants to make consistency fail. The consistency is formally defined through the experiment

$Exp^{cons}_{\mathscr{A}}(\lambda)$ as follows:

$(params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}) \leftarrow \mathsf{Setup}(\lambda, m)$;
$((w^S, L^S_l), (w^R_i, L^R_1, j, i)) \leftarrow \mathscr{A}(params, m)$ for $1 \leq l,\ 1 < j$ and $i \in [1,m]$;
$\mathscr{C}\mathscr{T}_{w^S} \leftarrow \mathsf{Encrypt}(params, w^S)$;
$Cert_{L^R_1,i} \leftarrow \mathsf{CertGen}(params, sk_C, L^R_1, i)$;
$UK_{L^R_j} \leftarrow \mathsf{UpdtKeyGen}^{(j-1)}(params, sk_C, AI_{L^R_1}, L^R_j, S_j)$;
$Cert_{L^R_j,i} \leftarrow \mathsf{UpdtCert}^{(j-1)}(params, sk_{R,i}, Cert_{L^R_1,i}, UK_{L^R_j})$;
$Trap_{w^R_i,L^R_j} \leftarrow \mathsf{TrapGen}(params, sk_R, w^R_i, Cert_{L^R_j,i})$;
If $i \in S_k$ for all $1 \leq k \leq j,\ (L^S_l = L^R_j),\ (w^S \neq w^R_i)$ and $\mathsf{Test}(params, sk_S, \mathscr{C}\mathscr{T}_{w^S}, L^S_l, Trap_{w^R_i,L^R_j}) \rightarrow 1$ then return $1$, else return $0$.

The advantage of $\mathscr{A}$ is defined as follows $Adv^{cons}_{\mathscr{A}}(\lambda) = Pr[Exp^{cons}_{\mathscr{A}}(\lambda) = 1]$. The scheme is said to be *computationally consistent* if any probabilistic polynomial time (PPT) adversary $\mathscr{A}$ wins the above experiment with negligible advantage.

**N.B.** The notation $\mathsf{UpdtKeyGen}^{(j-1)}$ denotes that the algorithm $\mathsf{UpdtKeyGen}$ is run $j-1$ times on inputs $L^R_2, L^R_3, \cdots, L^R_j$ respectively, and the notation $\mathsf{UpdtCert}^{(j-1)}$ denotes that the algorithm $\mathsf{UpdtCert}$ is run $j-1$ times on inputs $UK_{L^R_2}, UK_{L^R_3}, \cdots, UK_{L^R_j}$ respectively.

Jeong et al. [20] noticed that the consistency of a SCF-PEKS scheme turns this scheme to not be secure against keyword-guessing attacks, in case the adversary is the server. Nevertheless, as suggested in [12], this attack should not be considered. Instead, we regard the keyword-guessing attacks launched by an outsider (neither the server nor the certifier nor a receiver).

## 3.2 Indistinguishability of CBEKS against Chosen Keyword and Ciphertext attacks (IND-CKCA)

Let $\lambda$ be the security parameter, **KeywS** be the keyword space, **LabS** be the label space and **CiphS** be the ciphertext space. Let an adversary $\mathscr{A}$ and a challenger $\mathscr{B}$ play the following three games $Game_S$, $Game_C$ and $Game_R$.

### 3.2.1 Game played by the Server: $Game_S$.

The adversary $\mathscr{A}$ is assumed to be the server (inside attacker).

**Setup.** $\mathscr{B}$ runs the algorithm $\mathsf{Setup}$ on inputs $\lambda$ and $m$ to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}$. The challenger sends $params$ and $sk_S$ to $\mathscr{A}$.

**Query Phase 1.** $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L^R_1 >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L^R_1 \in$ **LabS** of its choice. The challenger answers by giving the first certificate $Cert_{L^R_1,i} \leftarrow \mathsf{CertGen}(params, sk_C, L^R_1, i)$ to $\mathscr{A}$ for which $i \in [1,m]$.

- *Update Key Query* $< L^R_j >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the update key query for any label $L^R_j \in$ **LabS**. The challenger first makes a first certificate query on $L^R_{j-1}$ to get $AI_{L^R_{j-1}}$, and answers

by giving the update key $UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j)$ to $\mathscr{A}$ for which $S_j \subseteq [1,m]$.

- *Refreshed Certificate Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the refreshed certificate query for any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on $L_{j-1}^R$ to get $Cert_{L_{j-1}^R,i}$ for which $i \in [1,m]$ and an update key query on $L_j^R$ to get $UK_{L_j^R}$, and answers by giving the refreshed certificate $Cert_{L_j^R} \leftarrow \mathsf{UpdtCert}(params, sk_{R,i}, UK_{L_j^R}, Cert_{L_{j-1}^R,i})$ to $\mathscr{A}$.

- *Trapdoor Query* $< w_i^R, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the trapdoor query for any keyword $w_i^R \in$ **KeywS** and any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query or a refreshed certificate query on $L_j^R$ to get $Cert_{L_j^R,i}$ for which $i \in [1,m]$, and answers by giving the trapdoor $Trap_{w_i^R,L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i})$ to $\mathscr{A}$.

- *Test Query* $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any ciphertext $\mathscr{CT}_{w^S} \in$ **CiphS**, any keyword $w_i^R \in$ **KeywS** and any labels $L_l^S, L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query or a refreshed query on $L_j^R$ to get $Cert_{L_j^R,i}$ and then a trapdoor query on $w_i^R$ to get $Trap_{w_i^R,L_j^R}$ for which $i \in [1,m]$, and answers by giving the result $b \leftarrow \mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R,L_j^R})$ to $\mathscr{A}$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a challenge keyword pair $(w_0, w_1)$ such that neither $w_0$ nor $w_1$ has been queried to obtain a corresponding trapdoor in the Query Phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge ciphertext $\mathscr{CT}_{w_\mu} \leftarrow \mathsf{Encrypt}(params, w_\mu)$. The challenger sends $\mathscr{CT}_{w_\mu}$ to the adversary. Note that the challenger randomly selects the bit $\mu$: we assume that the challenger cannot submit the same bit over and over.

**Query Phase 2.** $\mathscr{A}$ issues a number of queries as in the Query Phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > =< w_0, L_j^R >$ or $< w_i^R, L_j^R > =< w_1, L_j^R >$, and $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R > =< \mathscr{CT}_{w_0}, w_0, L_l^S, L_j^R >$ or $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R > =< \mathscr{CT}_{w_1}, w_1, L_l^S, L_j^R >$.

**Guess.** The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in $Game_S$ by $Adv_{\mathscr{A}}^{Game_S}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

### 3.2.2 Game played by the Certifier: $Game_C$.

The adversary $\mathscr{A}$ is assumed to be the certifier (outside attacker).

**Setup.** $\mathscr{B}$ runs the algorithm Setup on input $\lambda$ and $m$ to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}$. The challenger sends $params$ and $sk_C$ to $\mathscr{A}$.

**Query Phase 1.** $\mathscr{A}$ makes the queries as follows:

- *Refreshed Certificate Query* $< L_j^R >$. The same as in $Game_S$.

- *Trapdoor Query* $< w_i^R, L_j^R >$. The same as in $Game_S$, except that the certificate can only come from a refreshed certificate query.

- *Test Query* $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$. The same as in $Game_S$, except that the certificate can only come from a refreshed certificate query.

**Challenge.** The same as in $Game_S$.
**Query Phase 2.** The same as in $Game_S$.
**Guess.** The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.
We define the adversary's advantage in $Game_C$ by $Adv_{\mathscr{A}}^{Game_C}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

#### 3.2.3 Game played by the Receiver: $Game_R$.

The adversary $\mathscr{A}$ is assumed to be the receiver (outside attacker).
**Initialization.** $\mathscr{A}$ begins by selecting an index $i^* \in [1,m]$ of the receiver that it wants to play.
**Setup.** $\mathscr{B}$ runs the algorithm Setup on input $\lambda$ and $m$ to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}$. The challenger sends $params$ and $sk_{R,i^*}$ to $\mathscr{A}$.
**Query Phase 1.** $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L_1^R \in$ **LabS** of its choice. The challenger answers by giving the first certificate $Cert_{L_1^R,i^*} \leftarrow \mathsf{CertGen}(params, sk_C, L_1^R, i^*)$ to $\mathscr{A}$.

- *Update Key Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the update key query for any label $L_j^R \in$ **LabS**. The challenger first makes a first certificate query on $L_{j-1}^R$ to get $AI_{L_{j-1}^R}$, and answer by giving the update key $UK_{L_j^R} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j)$ to $\mathscr{A}$ for a group $S_j$ of receivers that includes $i^*$.

- *Test Query* $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any ciphertext $\mathscr{CT}_{w^S} \in$ **CiphS**, any keyword $w_{i^*}^R \in$ **KeywS** and any labels $L_l^S, L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on $L_j^R$ to get $Cert_{L_j^R,i^*}$ and generates $Trap_{w_{i^*}^R,L_j^R}$. and answers by giving the result $b \leftarrow \mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_{i^*}^R,L_j^R}, Cert_{L_j^R,i^*})$ to $\mathscr{A}$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a challenge keyword pair $(w_0, w_1)$ such that neither $w_0$ nor $w_1$ has been queried to obtain a corresponding trapdoor in the Query Phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge ciphertext $\mathscr{CT}_{w_\mu} \leftarrow \mathsf{Encrypt}(params, w_\mu)$. The challenger sends $\mathscr{CT}_{w_\mu}$ to the adversary. Note that the challenger randomly selects the bit $\mu$: we assume that the challenger cannot submit the same bit over and over.
**Query Phase 2.** $\mathscr{A}$ issues a number of queries as in the Query Phase 1. The restriction is that $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R > = < \mathscr{CT}_{w_0}, w_0, L_l^S, L_j^R >$ or $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R > = < \mathscr{CT}_{w_1}, w_1, L_l^S, L_j^R >$.
**Guess.** The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.
We define the adversary's advantage in $Game_R$ by $Adv_{\mathscr{A}}^{Game_R}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

**Definition 1.** *The CBEKS scheme is said to be IND-CKCA secure if* $Adv_{\mathscr{A}}^{Game_S}$, $Adv_{\mathscr{A}}^{Game_C}$ *and* $Adv_{\mathscr{A}}^{Game_R}$ *are all negligible.*

### 3.3 Indistinguishability of CBEKS against Keyword-Guessing attack (IND-KGA)

Let $\lambda$ be the security parameter, **KeywS** be the keyword space, **LabS** be the label space and **CiphS** be the ciphertext space. Let $\mathscr{A}$ be an outside adversary that is neither the server nor the certifier nor the receiver and that makes the keyword-guessing attack by interacting with a challenger $\mathscr{B}$. We consider the following game.

**Setup.** $\mathscr{B}$ runs the algorithm Setup on input $\lambda$ and $m$ to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}$. The challenger sends $params$ to $\mathscr{A}$.

**Query Phase 1.** $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. The same as in $Game_S$ from the IND-CKCA game.
- *Update Key Query* $< L_j^R >$. The same as in $Game_S$ from the IND-CKCA game.
- *Refreshed Certificate Query* $< L_j^R >$. The same as in $Game_S$ from the IND-CKCA game.
- *Trapdoor Query* $< w_i^R, L_j^R >$. The same as in $Game_S$ from the IND-CKCA game.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a challenge keyword pair $(w_0, w_1)$ such that neither $w_0$ nor $w_1$ has been queried to obtain a corresponding trapdoor in the Query Phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and a label $L_j^R$, and by computing a challenge trapdoor $Trap_{w_\mu, L_j^R} \leftarrow \mathsf{TrapGen}(params, sk_{R,i}, w_\mu, Cert_{L_j^R,i})$ where $Cert_{L_j^R,i}$ is the certificate issued for $L_j^R$. The challenger sends $Trap_{w_\mu, L_j^R}$ to the adversary. Note that the challenger randomly selects the bit $\mu$: we assume that the challenger cannot submit the same bit over and over.

**Query Phase 2.** $\mathscr{A}$ issues a number of queries as in the Query Phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R >=< w_0, L_j^R >$ or $< w_i^R, L_j^R >=< w_1, L_j^R >$.

**Guess.** The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in the above game by $Adv_{\mathscr{A}}^{IND-KGA}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

**Definition 2.** *The CBEKS scheme is said to be IND-KGA secure if* $Adv_{\mathscr{A}}^{IND-KGA}(\lambda)$ *is negligible.*

### 3.4 Collusion Resistance (CR)

Let $\lambda$ be the security parameter, **KeywS** be the keyword space, **LabS** be the label space and **CiphS** be the ciphertext space. Let $\mathscr{A}$ be a group of colluding receivers that attacks the collusion resistance of the update keys by interacting with a challenger $\mathscr{B}$. We consider the following game.

**Initialization.** $\mathscr{A}$ begins by selecting a group $S^* \subseteq [1,m]$ of receivers that it wants to be challenged on.

**Setup.** $\mathscr{B}$ runs the algorithm Setup on input $\lambda$ and $m$ to obtain $params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]}$. The challenger sends $params$ and $\{sk_{R,i}\}_{i\in[1,m]\setminus S^*}$ to $\mathscr{A}$.

**Query Phase 1.** $\mathscr{A}$ makes the queries as follows:

- *First Certificate Query* $< L_1^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the first certificate query for any label $L_1^R \in$ **LabS** of its choice. The challenger answers by giving the certificate $Cert_{L_1^R,i} \leftarrow \mathsf{CertGen}($ $params, sk_C, L_1^R, i)$ to $\mathscr{A}$ for which $i \in S \subseteq S^*$.
- *Refreshed Certificate Query* $< L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the refreshed certificate query for any label $L_j^R \in$ **LabS** of its choice. The challenger first makes a first certificate query on $L_{j-1}^R$ to get $Cert_{L_{j-1}^R,i}$ for which $i \in S \subseteq S^*$, then computes an update key $UK_{L_j^R}$ for $L_j^R$, and answers by giving the refreshed certificate $Cert_{L_j^R} \leftarrow \mathsf{UpdtCert}(params, sk_{R,i}, UK_{L_j^R}, Cert_{L_{j-1}^R,i})$ to $\mathscr{A}$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a challenge label pair $(L_0, L_1)$ such that neither $L_0$ nor $L_1$ has been queried to obtain a corresponding certificate or trapdoor in the Query Phase 1. Upon receiving this pair, $\mathscr{B}$ answers by choosing a random bit $\mu \in_R \{0,1\}$ and by computing a challenge update key $UK_{L_\mu} \leftarrow \mathsf{UpdtKeyGen}(params, sk_C, AI_{L_\mu - 1}, L_\mu, S^*)$. The challenger

sends $UK_{L_\mu}$ to the adversary. (We denote by $L_\mu - 1$ the label preceding the label $L_\mu$.) Note that the challenger randomly selects the bit $\mu$: we assume that the challenger cannot submit the same bit over and over.

**Query Phase 2.** $\mathscr{A}$ issues a number of queries as in the Query Phase 1. The restriction is that $< L_j^R >$ are not allowed to be queried as first certificate or refreshed certificate queries if $< L_j^R >=< L_0 >$ or $< L_j^R >=< L_1 >$.

**Guess.** The adversary output the guess $\mu' \in \{0,1\}$ and wins if $\mu' = \mu$.

We define the adversary's advantage in the above game by $Adv_{\mathscr{A}}^{CR}(\lambda) = |Pr[\mu' = \mu] - 1/2|$.

**Definition 3.** *The CBEKS scheme is said to be collusion resistant if $Adv_{\mathscr{A}}^{CR}(\lambda)$ is negligible.*

# 4 Preliminaries

## 4.1 Bilinear Maps

Let $\mathbb{BG}$ be an algorithm that on input a security parameter $\lambda$, outputs the parameters $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ for a bilinear mapping, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order $p$. Let $g_1$ be a generator of $\mathbb{G}_1$ and $g_2$ be a generator of $\mathbb{G}_2$. The function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if the following conditions hold:

1. *Bilinear:* $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$;
2. *Non-degenerate:* $e(g_1, g_2) \neq 1$;
3. *Efficiently computable:* There is an efficient algorithm that computes $e(h_1, h_2)$ for all $h_1 \in \mathbb{G}_1$ and $h_2 \in \mathbb{G}_2$.

## 4.2 Discrete Logarithm Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative groups of prime order $p$. Let $g_1$ be a generator of $\mathbb{G}_1$ and $g_2$ be a generator of $\mathbb{G}_2$. We define the advantage function $Adv_{\mathscr{B},\mathbb{G}_1}^{DL}(\lambda)$ of an adversary $\mathscr{B}$ as $Pr[\mathscr{B}(g_1, g_1^a) = a]$ where $a$ is randomly chosen in $\mathbb{Z}_p$. We similarly define the advantage function $Adv_{\mathscr{B},\mathbb{G}_2}^{DL}(\lambda)$ of an adversary $\mathscr{B}$ as $Pr[\mathscr{B}(g_2, g_2^a) = a]$ where $a$ is randomly chosen in $\mathbb{Z}_p$.

We say that the Discrete Logarithm (DL) assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if both $Adv_{\mathscr{B},\mathbb{G}_1}^{DL}(\lambda)$ and $Adv_{\mathscr{B},\mathbb{G}_2}^{DL}(\lambda)$ are negligible for all PPT adversary $\mathscr{B}$.

## 4.3 Symmetric External Diffie-Hellman Assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be the parameters for a bilinear mapping. We define the advantage function $Adv_{\mathscr{B},\mathbb{G}_1}^{SXDH}(\lambda)$ of an adversary $\mathscr{B}$ as

$$|Pr[\mathscr{B}(g_1, g_1^\theta, g_1^\omega, g_1^{\theta\omega}) = 1] - Pr[\mathscr{B}(g_1, g_1^\theta, g_1^\omega, g_1^\sigma) = 1]|$$

where $\theta, \omega, \sigma$ are randomly chosen in $\mathbb{Z}_p$. We also define the advantage function $Adv_{\mathscr{B},\mathbb{G}_2}^{SXDH}(\lambda)$ of an adversary $\mathscr{B}$ as

$$|Pr[\mathscr{B}(g_2, g_2^\theta, g_2^\omega, g_2^{\theta\omega}) = 1] - Pr[\mathscr{B}(g_2, g_2^\theta, g_2^\omega, g_2^\sigma) = 1]|$$

where $\theta, \omega, \sigma$ are randomly chosen in $\mathbb{Z}_p$.

We say that the Symmetric External Diffie-Hellman (SXDH) assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if both $Adv_{\mathscr{B},\mathbb{G}_1}^{SXDH}(\lambda)$ and $Adv_{\mathscr{B},\mathbb{G}_2}^{SXDH}(\lambda)$ are negligible for all PPT adversary $\mathscr{B}$. We set $Adv_{\mathscr{B},\mathbb{G}_1,\mathbb{G}_2}^{SXDH}(\lambda) \geq Adv_{\mathscr{B},\mathbb{G}_1}^{SXDH}(\lambda) + Adv_{\mathscr{B},\mathbb{G}_2}^{SXDH}(\lambda)$.

### 4.4 Decisional Bilinear Diffie-Hellman Assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be the parameters for a bilinear mapping. We define the advantage function $Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)$ of an adversary $\mathscr{B}$ as

$$|Pr[\mathscr{B}(g_1, g_1^{\theta}, g_1^{\delta}, g_2, g_2^{\delta}, g_2^{\omega}, e(g_1,g_2)^{\theta\delta\omega}) = 1] - Pr[\mathscr{B}(g_1, g_1^{\theta}, g_1^{\delta}, g_2, g_2^{\delta}, g_2^{\omega}, e(g_1,g_2)^{\sigma}) = 1]|$$

where $\theta, \delta, \omega, \sigma$ are randomly chosen in $\mathbb{Z}_p$.

We say that the Decisional Bilinear Diffie-Hellman Assumption (DBDH) assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ if $Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)$ is negligible for all PPT adversary $\mathscr{B}$.

### 4.5 Decisional Bilinear Diffie-Hellman Exponent Assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be the parameters for a bilinear mapping. We define the advantage function $Adv^{m-DBDHE}_{\mathscr{B},\mathbb{G}_T}(\lambda)$ of an adversary $\mathscr{B}$ as

$$\begin{aligned}&|Pr[\mathscr{B}(g_1, g_1^{\delta}, g_1^{\alpha}, \cdots, g_1^{\alpha^m}, g_1^{\alpha^{m+2}}, \cdots, g_1^{\alpha^{2m}}, g_2, g_2^{\delta}, g_2^{\alpha}, g_2^{\alpha^2}, \cdots, g_2^{\alpha^m}, e(g_1,g_2)^{\delta\alpha^{m+1}}) = 1] \\ &-Pr[\mathscr{B}(g_1, g_1^{\delta}, g_1^{\alpha}, \cdots, g_1^{\alpha^m}, g_1^{\alpha^{m+2}}, \cdots, g_1^{\alpha^{2m}}, g_2, g_2^{\delta}, g_2^{\alpha}, g_2^{\alpha^2}, \cdots, g_2^{\alpha^m}, e(g_1,g_2)^{\sigma}) = 1]|\end{aligned}$$

where $\alpha, \delta, \sigma$ are randomly chosen in $\mathbb{Z}_p$.

We say that the $m$-Decisional Bilinear Diffie-Hellman Exponent Assumption (DBDHE) assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ if $Adv^{m-DBDHE}_{\mathscr{B},\mathbb{G}_T}(\lambda)$ is negligible for all PPT adversary $\mathscr{B}$.

### 4.6 Waters Hash Function

We choose to base our CBEKS construction on Waters IBE scheme [29] since such scheme is efficient and practical, as well as taking place in the standard model. Indeed, Waters provided a hash function $H : \{0,1\}^n \to \mathbb{G}_1$ that is collision resistant, while is not seen as a random oracle.

Let $H : \{0,1\}^n \to \mathbb{G}_1$ be the hash function used in Waters IBE scheme [29]. First, pick at random $n+1$ exponents $e_0, e_1, \cdots, e_n \in_R \mathbb{Z}_p$ and then compute $h_i = g^{e_i}$ for $i \in [0,n]$. Let $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ be the public description of the hash function $H$.

In the next CBEKS construction, the algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as the product

$$h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i) \text{ and } H(w) = h_0 \cdot \prod_{i=1}^{n}(h_i^{w_i}) = g_1^{h(w)}.$$

The part $h(w)$ can be seen as the secret key that protects the keyword.

In [29], Waters compared his IBE scheme with the Boneh-Boyen (BB) IBE system [4] in terms of security. Let $u$ be a random element in $\mathbb{G}_1$ and $id$ be an identity. In BB scheme, $id$ is seen as an element in $\mathbb{Z}_p$ whereas in Waters scheme, $id = (id_1, \cdots, id_n) \in \{0,1\}^n$. Boneh and Boyen evaluated the element $u \cdot g_1^{id}$ while Waters considered the element $u \cdot \prod_{i \in \mathscr{S}} id_i$ where $\mathscr{S} \subset [1,n]$ is the set of all $i$ for which $id_i = 1$. This difference makes that Waters scheme is fully secure whereas BB scheme is only selectively secure. Therefore, we use Waters' trick to construct our CBEKS scheme to obtain the desired security level.

# 5 CBEKS Construction

The following CBEKS construction is inspired from the Boneh-Gentry-Waters (BGW) Broadcast Encryption (BE) scheme [6]. A BE scheme allows a sender to forward encrypted information to all the recipients such that only a group of recipients selected by the sender can recover the original information in plain. Such property is useful to let the certifier to send encrypted update keys to all the receivers, while only some of them are able to successfully retrieve these update keys regarding the selection of the certifier. Thus, we let the certifier choose a group of receivers when it is generating an update key, such that only the receivers in this group will be able to correctly update their certificate. Therefore, it seems natural that the construction will lead to BE schemes. The main advantage of the BGW scheme is the constant size of both the receiver's secret keys and the ciphertexts. The scheme is proved collusion resistant and selectively secure against chosen-ciphertext attacks in the standard model.

The construction below is also established on Waters IBE scheme [29], where the algorithm $\mathsf{KeyGen}$ in the IBE scheme corresponds to the algorithm $\mathsf{TrapGen}$ in our CBEKS scheme. We let each receiver choose a keyword and generate a corresponding trapdoor that is given to the server in order to check that the receiver's keyword matches the uploader's one. The Waters scheme is efficient in that the secret key and the ciphertext have constant size, and the decryption only involves two pairing computations. The scheme is proved semantically secure in the standard model. Observe that Abdalla et al. [1] showed that Waters IBE scheme is not anonymous (meaning that the ciphertext might reveal the identity of the recipient). Moreover, Boneh et al. [5] presented a transformation of an IBE scheme into a PEKS scheme. Nevertheless, the authors noticed that the IBE scheme is required to be anonymous in order to provide a PEKS scheme against chosen message attacks. Therefore, such issue directly applied to our CBEKS scheme; however we manage to overcome it as follows.

First, note that in addition to the keywords that have to match, a receiver should provide a label $L_j^R$ that corresponds to the label $L_l^S$ held by the server for a successful test outcome. This means that a receiver meets two verification steps through the label and the keyword that enhance the security of the CBEKS scheme.

We now assume that the label of the receiver $L_j^R$ matches the label of the server $L_l^S$. As noticed by Fang et al. [12], we have to ensure that an adversarial receiver cannot modify a ciphertext $\mathscr{CT}_{w^S}$ into a new valid ciphertext $\mathscr{CT}'_{w^S}$ without knowing the keyword $w^S$. However, this adversarial receiver would be able to generate a trapdoor $Trap_{w_i^R,L_j^R}$ for a guessed keyword $w_i^R$ using its secret key and so could obtain the relation between the modified ciphertext $\mathscr{CT}'_{w^S}$ and the trapdoor $Trap_{w_i^R,L_j^R}$ through interacting with the server as in a real environment. For this reason, Fang et al. [12] suggested to introduce a test query in the security model, as well as a strongly unforgeable one-time signature $\sigma = \mathsf{Sign}(ssk, (\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5))$ on the tuple $(\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5)$ such that $\mathscr{CT}_4 = g_1^{\kappa}$ and $\mathscr{CT}_5 = (A^{svk}B)^{\kappa}$ for a random exponent $\kappa \in_R \mathbb{Z}_p$ and a verification key $svk$.

Our CBEKS construction is as follows:

$\mathsf{Setup}(\lambda, m) \rightarrow (params, sk_S, sk_C, \{sk_{R,i}\}_{i\in[1,m]})$. Let $\lambda$ be the security parameter and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be the bilinear map parameters. Pick at random $A, B \in_R \mathbb{G}_2$ and let $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a strongly unforgeable one-time signature scheme. Pick at random $\alpha \in_R \mathbb{Z}_p$ and compute $g_1^{\alpha^i}$ for $i \in [1,m] \cup [m+2, 2m]$ and $g_2^{\alpha^i}$ for $i \in [1,m]$. Pick at random $\beta \in_R \mathbb{Z}_p$ and compute $g_1^{\beta}$ and $g_2^{\beta}$. Pick at random $\gamma \in_R \mathbb{Z}_p$ and compute $g_1^{\gamma\alpha^i}$ for $i \in [0,m]$. Pick at random $a_1, \cdots, a_m, b, c \in_R \mathbb{Z}_p$ and compute $g_1^{a_1}, \cdots, g_1^{a_m}$, $g_2^b$ and $g_1^c$.

Finally, set the public parameters *params* as equal to:

$$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,m]\cup[m+2,2m]}, \{g_2^{\alpha^i}\}_{i\in[1,m]}, g_2^{\beta}, \{g_1^{a_i}\}_{i\in[1,m]}, g_2^b, g_1^c, \mathsf{OTS}).$$

Set the receiver $i$'s secret key $sk_{R,i}$ as equal to $(a_i, g_1^{\beta}, g_1^{\gamma\alpha^i})$ for $i \in [1,m]$. Set the server's secret key $sk_S$ as equal to $b$. Set the certifier's secret key $sk_C$ as equal to $(c, g_1^{\gamma})$.

$\mathsf{Encrypt}(params, w^S) \rightarrow \mathscr{CT}_{w^S}$. Let the keyword space be $\mathbf{KeywS} = \{0,1\}^n$ where $2^n << p$. Choose $n+1$ random elements $e_0, e_1, \cdots, e_n \in_R \mathbb{Z}_p$ and compute $h_k = g_1^{e_k}$ for $k \in [0,n]$. Set $h = (h_0, h_1, \cdots, h_n)$ as the public description of the Waters hash function $H : \{0,1\}^n \rightarrow \mathbb{G}_1$, and $e = (e_0, e_1, \cdots, e_n)$ is kept secret by the uploader.

Second, select a one-time signature key pair $(ssk, svk) \leftarrow \mathsf{KeyGen}(\lambda)$. Pick at random $y, \kappa \in_R \mathbb{Z}_p$ and compute $g_1^{h(w^S)y}$, $(g_2^{\beta} \cdot g_2^{-w^S})^y = g_2^{(\beta - w^S)y}$, $(g_2^b)^y$, $g_1^{\kappa}$ and $(A^{svk}B)^{\kappa}$. Then, compute the one-time signature $\sigma = \mathsf{Sign}(ssk, (g_1^{h(w^S)y}, g_2^{(\beta-w^S)y}, g_2^{by}, g_1^{\kappa}, (A^{svk}B)^{\kappa}))$. Finally, set the ciphertext $\mathscr{CT}_{w^S}$ as equal to:

$$(\mathscr{CT}_0, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma) = (svk, g_1^{h(w^S)y}, g_2^{(\beta-w^S)y}, g_2^{by}, g_1^{\kappa}, (A^{svk}B)^{\kappa}, \sigma).$$

$\mathsf{CertGen}(params, sk_C, L_1^R, i) \rightarrow Cert_{L_1^R,i}$. Pick at random $r_1 \in_R \mathbb{Z}_p$ and compute $e(g_1^{a_i}, g_2^b)^{cr_1 L_1^R}$ and $g_2^{r_1}$. The certifier sends $(e(g_1^{a_i}, g_2^b)^{cr_1 L_1^R}, g_2^{r_1})$ to the receiver and the latter calculates $(g_2^{r_1})^{a_i}$ (where $a_i$ is one of the components of the receiver $i$'s secret key $sk_{R,i}$). Moreover, the certifier keeps $AI_{L_1^R} = g_1^{r_1 L_1^R}$ on its local storage. Finally, set the certificate $Cert_{L_1^R,i}$ as equal to:

$$(C_{1,1}, C_{1,2}) = (e(g_1^{a_i}, g_2^b)^{cr_1 L_1^R}, g_2^{a_i r_1}).$$

$\mathsf{UpdtKeyGen}(params, sk_C, AI_{L_{j-1}^R}, L_j^R, S_j) \rightarrow UK_{L_j^R}$. Let $AI_{L_{j-1}^R} = g_1^{r_{j-1} L_{j-1}^R}$ and $S_j \subseteq [1,m]$. Pick at random $s_j, r_j \in_R \mathbb{Z}_p$ and compute $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot \prod_{k \in S_j} g_1^{\alpha^{m+1-k}})^{s_j}$ and $e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot \frac{e(g_1^c, g_2^b)^{r_j L_j^R}}{e(g_1^{r_{j-1} L_{j-1}^R}, g_2^b)^c} = e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R}$. The certifier keeps $AI_{L_j^R} = g_1^{r_j L_j^R}$ on its local storage. Finally, set the update key $UK_{L_j^R}$ as equal to:

$$(uk_{j,1}, uk_{j,2}, uk_{j,3}, uk_{j,4}) = (g_2^{s_j}, g_2^{r_j}, (g_1^{\gamma} \cdot \prod_{k \in S_j} g_1^{\alpha^{m+1-k}})^{s_j}, e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R}).$$

$\mathsf{UpdtCert}(params, Cert_{L_{j-1}^R,i}, UK_{L_j^R}) \rightarrow Cert_{L_j^R,i}$. Suppose that $i \in S_j$. First, parse the certificate $Cert_{L_{j-1}^R,i}$ as $(C_{j-1,1}, C_{j-1,2}) = (e(g_1^{a_i}, g_2^b)^{cr_{j-1} L_{j-1}^R}, g_2^{a_i r_{j-1}})$ and the update key $UK_{L_j^R}$ as $(uk_{j,1}, uk_{j,2}, uk_{j,3}, uk_{j,4}) = (g_2^{s_j}, g_2^{r_j}, (g_1^{\gamma} \cdot \prod_{k \in S_j} g_1^{\alpha^{m+1-k}})^{s_j}, e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R})$. Second, compute:

$$\begin{aligned}
\frac{e(uk_{j,3}, g_2^{\alpha^i})}{e(g_1^{\gamma\alpha^i} \cdot \prod_{k \in S_j, k \neq i} g_1^{\alpha^{m+1-k+i}}, uk_{j,1})} &= \frac{e((g_1^{\gamma} \cdot \prod_{k \in S_j} g_1^{\alpha^{m+1-k}})^{s_j}, g_2^{\alpha_i})}{e(g_1^{\gamma\alpha^i} \cdot \prod_{k \in S_j, k \neq i} g_1^{\alpha^{m+1-k+i}}, g_2^{s_j})} \\
&= e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \\
\frac{uk_{j,4}}{e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j}} &= \frac{e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R}}{e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j}} \\
&= e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R}
\end{aligned}$$

Then, compute $(uk_{j,2})^{a_i} = (g_2^{r_j})^{a_i}$ (where $a_i$ is one of the components of the receiver $i$'s secret key $sk_{R,i}$) and

$$\begin{aligned}
C_{j-1,1} \cdot (e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R})^{a_i} &= e(g_1^{a_i}, g_2^b)^{cr_{j-1} L_{j-1}^R} \cdot (e(g_1^c, g_2^b)^{r_j L_j^R - r_{j-1} L_{j-1}^R})^{a_i} \\
&= e(g_1^{a_i}, g_2^b)^{cr_j L_j^R}
\end{aligned}$$

Finally, set the refreshed certificate $Cert_{L_j^R,i}$ as equal to:

$$(C_{j,1}, C_{j,2}) = (e(g_1^{a_i}, g_2^b)^{cr_j L_j^R}, g_2^{a_i r_j}).$$

$\mathsf{TrapGen}(params, sk_{R,i}, w_i^R, Cert_{L_j^R,i}) \rightarrow Trap_{w_i^R, L_j^R}$. First, parse the certificate $Cert_{L_j^R,i}$ as $(C_{j,1}, C_{j,2}) = (e(g_1^{a_i}, g_2^b)^{cr_j L_j^R}, g_2^{a_i r_j})$. Keep $C_{j,2} = g_2^{a_i r_j}$, pick at random $v, x, z \in_R \mathbb{Z}_p$ and compute $e(H(w_i^R), g_2)^{a_i x}$, $(g_2^b)^{a_i z}$, $g_1^v$ as well as

$$\begin{aligned} C_{j,1} \cdot e(H(w_i^R), g_2^b)^{a_i x} &= e(g_1^{a_i}, g_2^b)^{cr_j L_j^R} \cdot e(H(w_i^R), g_2^b)^{a_i x} \\ (g_1^\beta \cdot g_1^{-w_i^R})^v \cdot (H(w_i^R))^{a_i z} &= g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{a_i z} \end{aligned}$$

Finally, set the trapdoor $Trap_{w_i^R, L_j^R}$ as equal to:

$$\begin{aligned} (T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}) = {} & (e(g_1^{a_i}, g_2^b)^{cr_j L_j^R} \cdot e(H(w_i^R), g_2^b)^{a_i x}, e(H(w_i^R), g_2)^{a_i x}, \\ & g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{a_i z}, g_2^{b a_i z}, g_2^{a_i r_j}, g_1^v) \end{aligned}$$

$\mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R, L_j^R}) \rightarrow b$. First, parse the ciphertext $\mathscr{CT}_{w^S}$ as $(\mathscr{CT}_0, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma) = (svk, g_1^{h(w^S)y}, g_2^{(\beta - w^S)y}, g_2^{by}, g_1^\kappa, (A^{svk}B)^\kappa, \sigma)$ and the trapdoor $Trap_{w_i^R, L_j^R}$ as $(T_{j,1}, T_{j,2}, T_{j,3}, T_{j,4}, T_{j,5}, T_{j,6}) = (e(g_1^{a_i}, g_2^b)^{cr_j L_j^R} \cdot e(H(w_i^R), g_2^b)^{a_i x}, e(H(w_i^R), g_2)^{a_i x}, g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{a_i z}, g_2^{b a_i z}, g_2^{a_i r_j}, g_1^v)$.

Second, test if

$$\mathsf{Verify}(\mathscr{CT}_0, \sigma, (\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5)) = 1 \text{ and } e(\mathscr{CT}_4, A^{\mathscr{CT}_0} B) = e(g_1, \mathscr{CT}_5).$$

Then, check that

$$T_{j,1} = e(g_1^c, T_{j,5})^{bL_l^S} \cdot (T_{j,2})^b \text{ and } \frac{e(T_{j,3}, \mathscr{CT}_3)}{e(\mathscr{CT}_1, T_{j,4})} = e(T_{j,6}, \mathscr{CT}_2)^b.$$

If all the adove equations hold, then output 1; otherwise, output 0.

**Correctness** First, we verify that:

$$e(\mathscr{CT}_4, A^{\mathscr{CT}_0} B) = e(g_1^\kappa, A^{svk} B) = e(g_1, (A^{svk} B)^\kappa) = e(g_1, \mathscr{CT}_5)$$

Then, we suppose that $w^S = w^R$ and $L_l^S = L_j^R$ and we verify that:

$$\begin{aligned}
T_{j,1} &= e(g_1^{a_i}, g_2^b)^{cr_j L_j^R} \cdot e(H(w_i^R), g_2^b)^{a_i x} \\
&= e(g_1^c, g_2^{a_i r_j})^{bL_l^S} \cdot (e(H(w_i^R), g_2)^{a_i x})^b \\
&= e(g_1^c, T_{j,5})^{bL_l^S} \cdot (T_{j,2})^b \\
\frac{e(T_{j,3}, \mathscr{CT}_3)}{e(\mathscr{CT}_1, T_{j,4})} &= \frac{e(g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{a_i z}, g_2^{by})}{e(g_1^{h(w^S)y}, g_2^{ba_i z})} \\
&= e(g_1^{(\beta - w_i^R)v}, g_2^{by}) \cdot \frac{e(H(w_i^R)^{a_i z}, g_2^{by})}{e(g_1^{h(w^S)y}, g_2^{ba_i z})} \\
&= e(g_1^{(\beta - w_i^R)v}, g_2^{by}) \cdot \frac{e((g_1^{h(w_i^R)})^{a_i z}, g_2^{by})}{e(g_1^{h(w^S)y}, g_2^{ba_i z})} \\
&= e(g_1^{(\beta - w^S)v}, g_2^{by}) \cdot \frac{e((g_1^{h(w^S)})^{a_i z}, g_2^{by})}{e(g_1^{h(w^S)y}, g_2^{ba_i z})} \\
&= e(g_1^v, g_2^{(\beta - w^S)y})^b = e(T_{j,6}, \mathscr{CT}_2)^b
\end{aligned}$$

# 6 Security Proofs

## 6.1 Consistency

**Theorem 1.** *The CBEKS scheme is computationally consistent without the random oracle model assuming that the DL assumption holds.*

**Proof** Suppose there exists a PPT adversary $\mathscr{A}$ that attacks the computational consistency of the CBEKS scheme. A challenger $\mathscr{B}$ tries to solve the DL problem by playing the consistency game with the adversary $\mathscr{A}$ as follows.

$\mathscr{B}$ receives a DL problem instance $(g_1, g_1^{a_i}, g_2, g_2^{a_i})$ and has to output $a_i$. Let $Adv^{DL}_{\mathscr{B},\mathbb{G}_1}(\lambda)$ be the advantage function that $\mathscr{B}$ solves the DL problem in $\mathbb{G}_1$ and $Adv^{DL}_{\mathscr{B},\mathbb{G}_2}(\lambda)$ be the advantage function that $\mathscr{B}$ solves the DL problem in $\mathbb{G}_2$.

Let $(w^S, L_l^S)$ and $(w_i^R, L_1^R, j, i)$ be the tuples that $\mathscr{A}$ returns in the consistency experiment. Let a ciphertext $\mathscr{CT}_{w^S}$ be

$$(\mathscr{CT}_0 = svk, \mathscr{CT}_1 = g_1^{h(w^s)y}, \mathscr{CT}_2 = g_2^{(\beta - w^S)y}, \mathscr{CT}_3 = g_2^{by}, \mathscr{CT}_4 = g_1^{\kappa}, \mathscr{CT}_5 = (A^{svk}B)^{\kappa})$$

for a one-time signature key pair $(ssk, svk) \leftarrow \mathsf{KeyGen}(\lambda)$, two elements $A, B$ in $\mathbb{G}_2$ and two random exponents $y, \kappa$ in $\mathbb{Z}_p$. Let a trapdoor $Trap_{w_i^R, L_j^R}$ be

$$\begin{aligned}
&(T_{j,1} = e(g_1^{a_i}, g_2^b)^{cr_j L_j^R} \cdot e(H(w_i^R)^{a_i}, g_2^b)^x, T_{j,2} = e(H(w_i^R)^{a_i}, g_2)^x, T_{j,3} = g_1^{(\beta - w_i^R)v} \cdot (H(w_i^R)^{a_i})^z, \\
&T_{j,4} = (g_2^{a_i})^{bz}, T_{j,5} = (g_2^{a_i})^{r_j}, T_{j,6} = g_1^v)
\end{aligned}$$

for random exponents $b, c, r_j, v, x, z$ in $\mathbb{Z}_p$. We let $H(w_i^R)^{a_i}$ be computed as $(g_1^{a_i})^{h(w_i^R)}$.

We assume that the algorithm $\mathsf{CertGen}$ was run with input $L_1^R$ and that the algorithms $\mathsf{UpdtKeyGen}$ and $\mathsf{UpdtCert}$ were called $j-1$ times to obtain $Cert_{L_j^R, i}$ and then $Trap_{w_i^R, L_j^R}$ since the receiver $i$ is supposed to belong to $S_k$ for all $1 \leq k \leq j$. (In addition, we suppose that $L_l^R = L_j^R$; however, we do not need this hypthesis in our proof.)

We suppose that the uploader's keyword and the receiver's keyword differ, i.e. $w^S \neq w_i^R$. The adversary wins exactly when $\frac{e(T_{j,3},\mathscr{C}\mathscr{T}_3)}{e(\mathscr{C}\mathscr{T}_1,T_{j,4})} = e(T_{j,6},\mathscr{C}\mathscr{T}_2)^b$. Therefore, we get that:

$$\begin{aligned}
&\Leftrightarrow \quad \frac{e(g_1^{(\beta-w_i^R)v}\cdot H(w_i^R)^{a_i z}, g_2^{by})}{e(g_1^{h(w^S)y}, g_2^{ba_i z})} = e(g_1^v, g_2^{(\beta-w^S)y})^b \\
&\Leftrightarrow \quad e(g_1^{(\beta-w_i^R)v}, g_2^{by})\cdot \frac{e(H(w_i^R)^{a_i z}, g_2^{by})}{e(g_1^{h(w^S)y}, g_2^{ba_i z})} = e(g_1^v, g_2^{(\beta-w^S)y})^b \\
&\Leftrightarrow \quad (\beta-w_i^R)vby + (h(w_i^R)-h(w^S))a_i zby = (\beta-w^S)vby \mod p
\end{aligned}$$

We suppose that $b, y \neq 0 \mod p$ (the event that both $b$ and $y$ are equal to $0 \mod p$ happens with probability $1/p^2$), so that $(\beta-w_i^R)v + (h(w_i^R)-h(w^S))a_i z = (\beta-w^S)v \mod p$.

- *Case 1:* $h(w_i^R) = h(w^S) \neq 0 \mod p$. In this situation, we get that $(\beta-w_i^R)v = (\beta-w^S)v$, and so $w_i^R = w^S$ such that $v \neq 0 \mod p$ (with probability $1-1/p$). We thus obtain a contradiction as we have assumed that $w^S \neq w_i^R$.
- *Case 2:* $h(w_i^R) \neq h(w^S)$. In this situation, we get that $(h(w_i^R)-h(w^S))a_i z = v(\beta-w^S-\beta+w_i^R) \mod p$, and so $a_i = \frac{w_i^R-w^S}{h(w_i^R)-h(w^S)}\cdot\frac{v}{z} \mod p$, meaning that we have a solution to the DL problem.
- *Case 3:* $h(w) = 0 \mod p$ *for* $w = w_i^R \vee w^S$. Let us fix a keyword $w$. The exponents $e_0, e_1, \cdots, e_n$ are randomly selected in $\mathbb{Z}_p$. Note that the total number of possibilities for $w$ is $2^n$. Then, we have that the probability that $h(w) = 0 \mod p$ is equal to $2^n/p$.

Finally, if $i \in S_k$ for all $1 \leq k \leq j$, $L_l^S = L_j^R$, $w^S \neq w_i^R$ and $\mathsf{Test}(params, sk_S, \mathscr{C}\mathscr{T}_{w^S}, L_l^S, Trap_{w_i^R, L_j^R}) \rightarrow 1$, the advantage of $\mathscr{A}$ is upper bounded as follows:

$$Adv_{\mathscr{A}}^{cons}(\lambda) = Pr[Exp_{\mathscr{A}}^{cons}(\lambda) = 1] \leq \frac{1}{p^2} + \frac{1}{p} + \frac{2^n}{p} + Adv_{\mathscr{B},\mathbb{G}_1}^{DL}(\lambda) + Adv_{\mathscr{B},\mathbb{G}_2}^{DL}(\lambda).$$

## 6.2 Indistinguishability of CBEKS against Chosen Keyword and Ciphertext attacks (IND-CKCA)

**Theorem 2.** *The CBEKS scheme is IND-CKCA secure without the random oracle model assuming that the SXDH assumption holds and that* OTS *is a strongly unforgeable one-time signature scheme.*

**Proof** The proof of this theorem will result from the proofs of the three lemmas. These lemmas represent $Game_S$ (server), $Game_C$ (certifier) and $Game_R$ (receiver), respectively.

**Lemma 1.** *The CBEKS scheme is semantically secure against a chosen keyword and ciphertext attack in* $Game_S$ *without the random oracle model assuming that the SXDH assumption holds.*

Suppose that there exists a PPT adversary $\mathscr{A}$ in $Game_S$ that can attack the CBEKS scheme in the standard model with advantage $Adv_{\mathscr{A}}^{Game_S}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least $\varepsilon$ in solving the SXDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. $\mathscr{B}$ receives a random SXDH problem instance $(g_1, g_1^\theta, g_1^\omega, g_2, g_2^\theta, g_2^\omega)$ and $Z$ that is either $g_2^{\theta\omega}$ or a random element in $\mathbb{G}_2$.

**Setup.** $\mathscr{B}$ computes the public parameters *params* and the server's secret key $sk_S$ as follows.

First, $\mathscr{B}$ chooses at random $\alpha \in_R \mathbb{Z}_p$ and generates $g_1^{\alpha^i}$ for $i \in [1,m] \cup [m+2, 2m]$ and $g_2^{\alpha^i}$ for $i \in [1,m]$. It also randomly chooses $a_1, \cdots, a_m, b, c \in_R \mathbb{Z}_p$ and computes $g_1^{a_1}, \cdots, g_1^{a_m}, g_2^b, g_1^c$. In addition, the

challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,m]\cup[m+2,2m]}, \{g_2^{\alpha^i}\}_{i\in[1,m]}, g_2^{\theta}, \{g_1^{a_i}\}_{i\in[1,m]}, g_2^{b}, g_1^{c}, \mathsf{OTS})$ and the server's secret key $sk_S = b$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction and that $\mathscr{B}$ has all the necessary values to compute the secret key $sk_S$.

Let the keyword space **KeywS** be $\{0,1\}^n$. $\mathscr{B}$ chooses $n+1$ random elements $e_0, e_1, \cdots, e_n$ in $\mathbb{Z}_p$ and computes $h_i = g_1^{e_i}$ for $i \in [0,n]$. Let $h = (h_0, h_1, \cdots, h_n)$ be the public description of the hash function $H$. The algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as $h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i)$ and $H(w) = h_0 \cdot \prod_{i=1}^{n}(h_i^{w_i}) = g_1^{h(w)}$.

**Query Phase 1.** $\mathscr{A}$ makes the following queries:

- *First Certificate Query* $< L_1^R >$. If $\mathscr{A}$ queries $L_1^R$ to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{a_i}, g_2^{b})^{cr_1 L_1^R}$ and $g_2^{r_1 a_i}$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R, i}$.

- *Update Key Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the update key generation oracle, then $\mathscr{B}$ randomly selects $\gamma, s_j, r_j$ in $\mathbb{Z}_p$ and generates $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot \prod_{k\in S} g_1^{\alpha^{m+1-k}})^{s_j}$ for $S \subseteq [1,m]$, and $e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^{c}, g_2^{b})^{r_j L_j^R - r_1 L_1^R}$. The challenger forwards these elements to $\mathscr{A}$ as the update key $UK_{L_j^R}$.

- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{a_i}, g_2^{b})^{cr_j L_j^R}$ and $g_2^{a_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R, i}$.

- *Trapdoor Query* $< w_i^R, L_j^R >$. If $\mathscr{A}$ queries $(w_i^R, L_j^R)$ to the trapdoor generation oracle, then $\mathscr{B}$ selects $v, x, z$ at random in $\mathbb{Z}_p$, generates $e(g_1^{a_i}, g_2^{b})^{cr_j L_j^R} \cdot e(H(w_i^R), g_2^{b})^{a_i x}$, $e(H(w_i^R), g_2)^{a_i x}$, $(g_1^{\theta})^{v} \cdot g_1^{-w_i^R v} \cdot H(w_i^R)^{a_i z}$, $(g_2^{b})^{a_i z}$, $g_2^{a_i r_j}$ and $g_1^{v}$ as the trapdoor $Trap_{w_i^R, L_j^R}$, and gives these elements to $\mathscr{A}$

- *Test Query* $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any $\mathscr{CT}_{w^S}$, $w_i^R$, $L_l^S$ and $L_j^R$. The challenger first makes a refreshed certificate query on $L_j^R$, then makes a trapdoor query on $w_i^R$ and $L_j^R$, and responds to $\mathscr{A}$ by sending the result $\mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R, L_j^R})$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a keyword pair $(w_0, w_1)$. The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by letting the challenge keyword be $w^* = w_\mu$. Then, it selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathsf{KeyGen}(\lambda)$ and an exponent $\kappa \in_R \mathbb{Z}_p$, and sets $\mathscr{CT}_0 = svk^*$, $\mathscr{CT}_4 = g_1^{\kappa}$ and $\mathscr{CT}_5 = (A^{svk^*} B)^{\kappa}$. It also sets $\mathscr{CT}_1 = (g_1^{\omega})^{h(w^*)}$, $\mathscr{CT}_2 = Z \cdot (g_2^{\omega})^{(-w^*)}$ and $\mathscr{CT}_3 = (g_2^{\omega})^{b}$, and generates a one-time signature $\sigma = \mathsf{Sign}(ssk^*, (\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5))$. The challenger sends the challenge ciphertext $\mathscr{CT}^* = (\mathscr{CT}_0, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma)$ to the adversary.

When $Z = g_2^{\theta\omega}$, then $\mathscr{CT}^*$ is a valid challenge ciphertext to $\mathscr{A}$ as in the real attack. When $Z$ is random in $\mathbb{G}_2$, then $\mathscr{CT}_2 = Z \cdot (g_2^{\omega})^{(-w^*)}$ is a uniform element in $\mathbb{G}_2$, and thus the ciphertext gives no information about the challenger's bit $\mu$.

**Query Phase 2.** $\mathscr{A}$ continues to make queries as in the Query Phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > = < w_0, L_j^R >$ or $< w_i^R, L_j^R > = <$

$w_1, L_j^R >$ and $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R > =< \mathscr{CT}^*, w_0, L_l^S, L_j^R >$ or $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R > =< \mathscr{CT}^*, w_1, L_l^S, L_j^R >$.

**Guess.** The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu = \mu'$, then $\mathscr{B}$ outputs 1 meaning that $Z = g_2^{\theta\omega}$; otherwise, $\mathscr{B}$ outputs 0 meaning that $Z$ is a random element in $\mathbb{G}_2$.

**Analysis** When $Z = g_2^{\theta\omega}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_2$, then $\mathscr{CT}_2 = Z \cdot (g_2^{\omega})^{(-w^*)}$ is uniformly random in $\mathbb{G}_2$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv^{SXDH}_{\mathscr{B},\mathbb{G}_1,\mathbb{G}_2}(\lambda) \geq \varepsilon$.

**Lemma 2.** *The CBEKS scheme is semantically secure against a chosen keyword and ciphertext attack in $Game_C$ without the random oracle model assuming that the SXDH assumption holds.*

Suppose that there exists a PPT adversary $\mathscr{A}$ in $Game_C$ that can attack the CBEKS scheme in the standard model with advantage $Adv^{Game_C}_{\mathscr{A}}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least $\varepsilon$ in solving the SXDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. $\mathscr{B}$ receives a random SXDH problem instance $(g_1, g_1^{\theta}, g_1^{\omega}, g_2, g_2^{\theta}, g_2^{\omega})$ and $Z$ that is either $g_2^{\theta\omega}$ or a random element in $\mathbb{G}_2$.

**Setup.** The Setup phase is similar to the one for $Game_S$ except the following:

1. $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes $g_2^{\beta}$;
2. $g_2^{\theta} = g_2^b$ for an unknown exponent $b$;
3. The certifier's secret key is set as $sk_C = (c, g_1^{\gamma})$ (the server's secret key $sk_S$ is not computed).

Note that $\mathscr{B}$ has all the necessary values to compute the secret key $sk_C$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,m]\cup[m+2,2m]}, \{g_2^{\alpha^i}\}_{i\in[1,m]}, g_2^{\beta}, \{g_1^{a_i}\}_{i\in[1,m]}, g_2^{\theta}, g_1^{c}, \mathsf{OTS})$ and the certifier's secret key $sk_C$.

**Query Phase 1.** $\mathscr{A}$ makes the following queries:

- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{a_i}, g_2^{\theta})^{cr_j L_j^R}$ and $g_2^{a_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R, i}$.
- *Trapdoor Query* $< w_i^R, L_j^R >$. If $\mathscr{A}$ queries $(w_i^R, L_j^R)$ to the trapdoor generation oracle, then $\mathscr{B}$ selects $v, x, z$ at random in $\mathbb{Z}_p$, generates $e(g_1^{a_i}, g_2^{\theta})^{cr_j L_j^R} \cdot e(H(w_i^R), g_2^{\theta})^{a_i x}$, $e(H(w_i^R), g_2)^{a_i x}$, $g_1^{(\beta - w_i^R)v} \cdot H(w_i^R)^{a_i z}$, $(g_2^{\theta})^{a_i z}$, $g_2^{a_i r_j}$ and $g_1^{v}$ as the trapdoor $Trap_{w_i^R, L_j^R}$, and gives these elements to $\mathscr{A}$.
- *Test Query* $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any $\mathscr{CT}_{w^S}$, $w_i^R$, $L_l^S$ and $L_j^R$. The challenger first makes a refreshed certificate query on $L_j^R$, then makes a trapdoor query on $w_i^R$ and $L_j^R$, and responds to $\mathscr{A}$ by sending the result $\mathsf{Test}(params, sk_S, \mathscr{CT}_{w^S}, L_l^S, Trap_{w_i^R, L_j^R})$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a keyword pair $(w_0, w_1)$. The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by letting the challenge keyword be $w^* = w_{\mu}$. Then, it selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathsf{KeyGen}(\lambda)$ and an exponent $\kappa \in_R \mathbb{Z}_p$, and sets $\mathscr{CT}_0 = svk^*$, $\mathscr{CT}_4 = g_1^{\kappa}$ and $\mathscr{CT}_5 = (A^{svk^*}B)^{\kappa}$. It also sets $\mathscr{CT}_1 = (g_1^{\omega})^{h(w^*)}$, $\mathscr{CT}_2 = (g_2^{\omega})^{(\beta - w^*)}$ and $\mathscr{CT}_3 = Z$, and generates a one-time signature $\sigma = \mathsf{Sign}(ssk^*, (\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5))$. The challenger sends the challenge ciphertext $\mathscr{CT}^* = (\mathscr{CT}_0, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma)$ to the adversary.

When $Z = g_2^{\theta\omega}$, then $\mathscr{CT}^*$ is a valid challenge ciphertext to $\mathscr{A}$ as in the real attack. When $Z$ is random in $\mathbb{G}_2$, then $\mathscr{CT}_3 = Z$ is a uniform element in $\mathbb{G}_2$, and so $\mathscr{CT}^*$ gives no information about the challenger's bit $\mu$.

**Query Phase 2.** $\mathscr{A}$ continues to make queries as in the Query Phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R > = < w_0, L_j^R >$ or $< w_i^R, L_j^R > = < w_1, L_j^R >$ and $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R > = < \mathscr{CT}^*, w_0, L_l^S, L_j^R >$ or $< \mathscr{CT}_{w^S}, w_i^R, L_l^S, L_j^R > = < \mathscr{CT}^*, w_1, L_l^S, L_j^R >$.

**Guess.** The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = g_2^{\theta\omega}$; otherwise, $\mathscr{B}$ outputs 0 meaning that $Z$ is a random element in $\mathbb{G}_2$.

**Analysis** When $Z = g_2^{\theta\omega}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_2$, then $\mathscr{CT}_3 = Z$ is uniformly random in $\mathbb{G}_2$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv^{SXDH}_{\mathscr{B},\mathbb{G}_1,\mathbb{G}_2}(\lambda) \geq \varepsilon$.

**Lemma 3.** *The CBEKS scheme is semantically secure against a chosen keyword and ciphertext attack in $Game_R$ without the random oracle model assuming that the SXDH assumption holds and that* OTS *is a strongly unforgeable one-time signature scheme.*

Suppose that there exists a PPT adversary $\mathscr{A}$ in $Game_R$ that can attack the CBEKS scheme in the standard model with advantage $Adv^{Game_R}_{\mathscr{A}}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least $\varepsilon$ in solving the SXDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$. $\mathscr{B}$ receives a random SXDH problem instance $(g_1, g_1^\theta, g_1^\omega, g_2, g_2^\theta, g_2^\omega)$ and $Z$ that is either $g_2^{\theta\omega}$ or a random element in $\mathbb{G}_2$.

Let $\mathscr{CT}^* = (svk^*, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma)$ be the challenge ciphertext given to the adversary in $Game_R$. Let $EvOTS$ be the event that $\mathscr{A}$ makes a test query for $\mathscr{CT} = (svk^*, \mathscr{CT}'_1, \mathscr{CT}'_2, \mathscr{CT}'_3, \mathscr{CT}'_4, \mathscr{CT}'_5, \sigma')$ such that $\mathsf{Verify}(svk^*, \sigma', (\mathscr{CT}'_1, \mathscr{CT}'_2, \mathscr{CT}'_3, \mathscr{CT}'_4, \mathscr{CT}'_5)) = 1$. In the Query Phase 1, the adversary does not have any information on $svk^*$. Thus, the probability of a pre-challenge occurrence of $EvOTS$ does not exceed $q_{to} \cdot Bound$ where $q_{to}$ denotes the total number of queries made to the test oracle and $Bound$ is the maximum probability that any one-time verification key $svk^*$ is output by KeyGen (which does not exceed $1/p$ by assumption). In the Query Phase 2, $EvOTS$ produces an algorithm that breaks the strong unforgeability of the one-time signature. Thus, the probability $Pr[EvOTS] \leq q_{to}/p + Adv^{OTS}$, where $Adv^{OTS}$ denotes the probability defined for a one-time signature (that should be negligible by assumption).

**Initialization.** $\mathscr{A}$ selects a receiver $i^* \in [1,m]$ as the one it wants to be challenged on.

**Setup.** The Setup phase is similar to the one for $Game_S$ except the following:

1. $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes $g_2^\beta$;
2. $g_2^\theta = g_2^b$ for an unknown exponent $b$;
3. The receiver $i^*$'s secret key is set as $sk_{R,i^*} = (a_{i^*}, g_1^\beta, g_1^{\gamma\alpha^{i^*}})$ (the server's secret key $sk_S$ is not computed).

Note that $\mathscr{B}$ has all the necessary values to compute the secret key $sk_{R,i^*}$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,m]\cup[m+2,2m]}, \{g_2^{\alpha^i}\}_{i\in[1,m]}, g_2^\beta, \{g_1^{a_i}\}_{i\in[1,m]}, g_2^\theta, g_1^c, \mathsf{OTS})$ and the receiver $i^*$'s secret key $sk_{R,i^*}$.

**Query Phase 1.** $\mathscr{A}$ makes the following queries:

- *First Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{a_{i^*}}, g_2^\theta)^{cr_1 L_1^R}$ and $g_2^{r_1 a_{i^*}}$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R, i^*}$.

- *Update Key Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the update key generation oracle, then $\mathscr{B}$ randomly selects $s_j, r_j$ in $\mathbb{Z}_p$ and generates $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot g_1^{\alpha^{m+1-i^*}})^{s_j}$, and $e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^c, g_2^{\theta})^{r_j L_j^R - r_1 L_1^R}$. The challenger forwards these elements to $\mathscr{A}$ as the update key $UK_{L_j^R}$.

- *Test Query* $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$. $\mathscr{A}$ can adaptively ask $\mathscr{B}$ for the test query for any $\mathscr{CT}_{w^S} = (\mathscr{CT}'_0, \mathscr{CT}'_1, \mathscr{CT}'_2, \mathscr{CT}'_3, \mathscr{CT}'_4, \mathscr{CT}'_5, \sigma')$, $w_{i^*}^R$, $L_l^S$ and $L_j^R$. The challenger first makes a first certificate query on $L_j^R$, then computes the trapdoor $Trap_{w_{i^*}^R, L_j^R}$, and tests if $\mathsf{Verify}(\mathscr{CT}'_0, \sigma', (\mathscr{CT}'_1, \mathscr{CT}'_2, \mathscr{CT}'_3, \mathscr{CT}'_4, \mathscr{CT}'_5)) = 1$ and $e(\mathscr{CT}'_4, A^{\mathscr{CT}'_0} B) = e(g_1, \mathscr{CT}'_5)$. If the above equations hold and given $\mathscr{CT}_{w^S} = (\mathscr{CT}'_0, \mathscr{CT}'_1, \mathscr{CT}'_2, \mathscr{CT}'_3, \mathscr{CT}'_4, \mathscr{CT}'_5, \sigma')$ and $\mathscr{CT}^* = (\mathscr{CT}_0, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma)$, then the challenger meets two cases:

  1. If $\mathscr{CT}'_0 = svk' \equiv svk^* = \mathscr{CT}_0$, then we get that the tuples $(\mathscr{CT}'_1, \mathscr{CT}'_2, \mathscr{CT}'_3, \mathscr{CT}'_4, \mathscr{CT}'_5, \sigma')$ and $(\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma)$ are not equal. Thus, the challenger sees an occurrence of the event $EvOTS$ and aborts.
  2. If $\mathscr{CT}'_0 = svk' \neq svk^* = \mathscr{CT}_0$, then we get that $e(\mathscr{CT}'_4, A^{\mathscr{CT}'_0} B) = e(g_1, \mathscr{CT}'_5)$ such that $\mathscr{CT}'_5 = (A^{svk'} B)^{\kappa}$ since the ciphertext is supposed to be valid.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a keyword pair $(w_0, w_1)$. The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by letting the challenge keyword be $w^* = w_\mu$. Then, it selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathsf{KeyGen}(\lambda)$ and an exponent $\kappa \in_R \mathbb{Z}_p$, and sets $\mathscr{CT}_0 = svk^*$, $\mathscr{CT}_4 = g_1^{\kappa}$ and $\mathscr{CT}_5 = (A^{svk^*} B)^{\kappa}$. It also sets $\mathscr{CT}_1 = (g_1^{\omega})^{h(w^*)}$, $\mathscr{CT}_2 = (g_2^{\omega})^{(\beta - w^*)}$ and $\mathscr{CT}_3 = Z$, and generates a one-time signature $\sigma = \mathsf{Sign}(ssk^*, (\mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5))$. The challenger sends the challenge ciphertext $\mathscr{CT}^* = (\mathscr{CT}_0, \mathscr{CT}_1, \mathscr{CT}_2, \mathscr{CT}_3, \mathscr{CT}_4, \mathscr{CT}_5, \sigma)$ to the adversary.

When $Z = g_2^{\theta\omega}$, then $\mathscr{CT}^*$ is a valid challenge ciphertext to $\mathscr{A}$ as in the real attack. When $Z$ is random in $\mathbb{G}_2$, then $\mathscr{CT}_3 = Z$ is a uniform element in $\mathbb{G}_2$, and thus the ciphertext gives no information about the challenger's bit $\mu$.

**Query Phase 2.** $\mathscr{A}$ continues to make queries as in the Query Phase 1. The restriction is that $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R >$ are not allowed to be queried as test queries if $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R > = < \mathscr{CT}^*, w_0, L_l^S, L_j^R >$ or $< \mathscr{CT}_{w^S}, w_{i^*}^R, L_l^S, L_j^R > = < \mathscr{CT}^*, w_1, L_l^S, L_j^R >$.

**Guess.** The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = g_2^{\theta\omega}$; otherwise, $\mathscr{B}$ outputs 0 meaning that $Z$ is a random element in $\mathbb{G}_2$.

**Analysis** If the event $EvOTS$ does not occur and when $Z = g_2^{\theta\omega}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_2$, then $\mathscr{CT}_3 = Z$ is uniformly random in $\mathbb{G}_2$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv_{\mathscr{B}, \mathbb{G}_1, \mathbb{G}_2}^{SXDH}(\lambda) \geq \varepsilon + \frac{q_{to}}{p} + Adv^{OTS}$.

### 6.3 Indistinguishability of CBEKS against Keyword-Guessing attack (IND-KGA)

**Theorem 3.** *The CBEKS scheme is IND-KGA secure without the random oracle model assuming that the DBDH assumption holds.*

**Proof** Suppose that there exists a PPT adversary $\mathscr{A}$ that can trigger a keyword-guessing attack against the CBEKS scheme in the standard model with advantage $Adv_{\mathscr{A}}^{IND-KGA}(\lambda)$. We build a challenger $\mathscr{B}$ that plays the DBDH game in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ by interacting with the adversary. $\mathscr{B}$ receives a random DBDH

problem instance $(g_1, g_1^{\theta}, g_1^{\delta}, g_2, g_2^{\delta}, g_2^{\omega}, Z)$ and outputs a bit $\nu' \in \{0,1\}$ as a guess to decide whether $Z$ is either equal to $e(g_1,g_2)^{\theta\delta\omega}$ or a random element in $\mathbb{G}_T$.

The proof is divided into four games, namely Game 1, Game 2, Game 3 and Game 4, that differ by slight modifications. In each game, the challenger will output a bit $\nu'$ that is well defined. Let $\mathscr{G}_i$ be the event that the adversary is successful in Game $i$, for $i \in [1,4]$. Such process will allow us to conclude.

**Game 1** This game is simply the same than the original IND-KGA security game. In the following, we describe the simulation of the challenger.

At the start of Game 1, $\mathscr{B}$ chooses three random exponents $\theta$, $\delta$ and $\omega$ uniformly in $\mathbb{Z}_p$. Nevertheless, in the next games, $\mathscr{B}$ will stop to use $\theta$, $\delta$ and $\omega$; instead, it will require $g_1^{\theta}, g_1^{\delta}, g_2^{\delta}, g_2^{\omega}$ along with either $Z = e(g_1,g_2)^{\theta\delta\omega}$ or $Z \in_R \mathbb{G}_T$ for the security game simulation.

The adversary and the challenger play Game 1 following several steps as in the definition of the IND-KGA security game. First, the challenger generates the public parameters *params* and gives them to $\mathscr{A}$. Then, the adversary initiates the Query Phase 1 and has access to the various oracles: $\mathscr{A}$ can make first certificate, update key, refreshed certificate and trapdoor queries. After this first phase of queries, the adversary chooses a challenge keyword pair $(w_0, w_1)$. The challenger chooses a bit $\mu \in_R \{0,1\}$ at random and sets the challenge keyword as $w^* = w_\mu$. Thereafter, $\mathscr{A}$ makes other queries to the same oracles such that the trapdoors resulting from trapdoor queries have to embed a keyword that differs from $w^*$. Eventually, the adversary returns a bit $\mu' \in \{0,1\}$ as a guess for $\mu$. If $\mu' = \mu$, then $\mathscr{B}$ returns $\nu' = 1$; otherwise, it returns $\nu' = 0$. This completes the description of the challenger's simulation in Game 1.

We give now more details about the trapdoor queries that $\mathscr{A}$ makes, for some keyword $w$ and label $L$. Let $q \in \mathbb{N}$ be the total number of trapdoor queries. Let $\bar{\mathscr{W}}$ be the set containing all the keywords selected for the trapdoor queries. We assume that the challenge keyword does not belong to $\bar{\mathscr{W}}$ since the restriction during the Challenge phase specifies that $w^*$ has not been and will not be queried in the Query Phase 1 and Phase 2, respectively. Let $\mathscr{W} \subseteq \bar{\mathscr{W}}$ be the subset of the requested keywords such that all multiples from $\bar{\mathscr{W}}$ are removed. We suppose that $|\mathscr{W}| = q_0 \leq q = |\bar{\mathscr{W}}|$ such that two keywords in $\mathscr{W}$ are necessarily different. Finally, we define $\mathscr{W}^* = \mathscr{W} \cup \{w^*\}$.

**Game 2** This game is almost identical to Game 1 except that the challenger sets some elements differently.

The challenger computes $M = 2q$ and randomly chooses $k \in_R [1,n]$. $\mathscr{B}$ then selects at random a tuple $x = (x_0, x_1, \cdots, x_n)$ where $x_i \in_R [0, M-1]$ for $i \in [1,n]$, and a tuple $y = (y_0, y_1, \cdots, y_n)$ where $y_i \in_R \mathbb{Z}_p$ for $i \in [1,n]$. We assume that these elements are kept secret by the challenger.

Given a keyword of the form $w = (w_1, \cdots, w_n)$, let us define three functions as follows:

$$x(w) = x_0 + \sum_{i=1}^{n}(x_i \cdot w_i),\ y(w) = (p - Mk) + y_0 + \sum_{i=1}^{n}(y_i \cdot w_i),\ h(w) = x(w) + \theta y(w).$$

The challenger generates the hash function $H$ with public description $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ as follows. $\mathscr{B}$ lets $h_0 = g_1^{x_0}(g_1^{\theta})^{p-Mk+y_0}$ and $h_i = g_1^{x_i}(g_1^{\theta})^{y_i}$ for $i \in [1,n]$. Observe that such setting does not affect the distribution of the outputs of the hash function $H$.

We now describe techniques employed in [29, 12] to construct the security proof. As in [12], let $VIEW_{\mathscr{A}}$ be the adversary's random tape and the transcript of its interactions with its oracles in the simulation of Game 2. In other words, let us fix all the random elements that $\mathscr{A}$ is able to learn during its execution, including its random coin tosses. More precisely, we fix the public parameters *params*, the

challenge bit $\mu$ and the randomness used in answering the trapdoor and other queries. This means that $\mathscr{A}$ can be seen as a deterministic algorithm, and thus the set $\mathscr{W}^*$ can be seen as fixed.

Let $\bar{\mathscr{Y}} = (y_0, y_1, \cdots, y_n, k)$ where the elements are distributed as above. Therefore, if $VIEW_{\mathscr{A}}$ is fixed and the game is conducted again, then $\bar{\mathscr{Y}}$ has the same distribution as for a run of the game without having $VIEW_{\mathscr{A}}$ as fixed. This happens due to the random "masking" values $x_i$.

*Forced Abort:* Let *FAbort* be the event that one of the following conditions is true:

1. $\mathscr{A}$ asks a trapdoor query for a keyword $w$ and a label $L$ such that $y(w) = 0 \mod p$.
2. $\mathscr{A}$ chooses a keyword pair $(w_0, w_1)$ such that neither $w_0$ nor $w_1$ is equal to $0 \mod p$.

If *FAbort* occurs then the challenger aborts and $\nu'$ is chosen at random.

We will modify the next games in order to force the challenger to abort each time that the above event happens (so we call the event a forced abort). For every fixed $VIEW_A$, we define $\tau(VIEW_{\mathscr{A}}) = Pr_{\bar{\mathscr{Y}}}[\neg FAbort]$, where $\neg FAbort$ denotes the complementary event of *FAbort*.

Let $\zeta_{low}$ and $\zeta_{up}$ be the lower and upper bounds on $\tau(VIEW_{\mathscr{A}})$ respectively. We get the following lemma:

**Lemma 4.** *For every fixed $VIEW_{\mathscr{A}}$, we define $\zeta_{low} = \frac{1}{4(n+1)q}$ and $\zeta_{up} = \frac{1}{2q}$. Thus, we have that*

$$\zeta_{low} \leq \tau(VIEW_{\mathscr{A}}) \leq \zeta_{up}.$$

This lemma is also (partly) used in [21, 29, 12]. Note that a proof of these two bounds can be found in [12]. We let the reader to refer to [12] for additional information about this lemma and the corresponding proof.

We now explicit the modifications made between Game 1 and Game 2. We assume that $VIEW_{\mathscr{A}}$ is fixed since we assume that the adversary has terminated the execution. Two events occuring in Game 2 but not in Game 1 can be described as follows:

*Forced Abort:* After the output of the adversary's bit $\mu'$ as a guess for $\mu$, check whether *FAbort* happens or not. If it occurs, then a random bit $\nu'$ is returned and the challenger aborts; otherwise, the challenger keeps on as before.

*Artificial Abort:* To discard some unwanted dependency on probabilities, some artificial aborts are added such that the challenger always aborts with probability approximately equal to $1 - \zeta_{low}$ and independently of $VIEW_{\mathscr{A}}$. More precisely, after the output of the adversary's bit $\mu'$ as a guess for $\mu$, check whether *FAbort* happens or not. If it occurs, then a random bit $\nu'$ is returned and the challenger aborts; otherwise, the challenger keeps on as follows. $\mathscr{B}$ first samples an estimate $\tau'(VIEW_{\mathscr{A}})$ of the probability $\tau(VIEW_{\mathscr{A}})$ that *FAbort* does not occur. Remember that $VIEW_{\mathscr{A}}$ is fixed now, thus the sampling does not involve running the adversary again. This estimate $\tau'(VIEW_{\mathscr{A}})$ is defined as a random variable and only depends on the keywords belonging to $\mathscr{W}^*$ and the randomness used to sample.

We now explicit the two cases that the challenger can encounter:

1. If $\tau'(VIEW_{\mathscr{A}}) \leq \zeta_{low}$, then we assume that $\mathscr{B}$ keeps on as before.
2. If $\tau'(VIEW_{\mathscr{A}}) > \zeta_{low}$, then the challenger aborts and outputs a bit $\nu'$ with probability equal to $1 - \frac{\zeta_{low}}{\tau'(VIEW_{\mathscr{A}})}$. (In other words, the challenger does not abort and keeps on as before with probability equal to $\frac{\zeta_{low}}{\tau'(VIEW_{\mathscr{A}})}$.)

The description of Game 2 is now completed.

The following claim is postulated in [21] and proved by Fang et al. [12]. It enables to bound the probabilities of $\mathscr{G}_1$ and $\mathscr{G}_2$:

**Claim 1.** *We define* $\rho(\lambda) \equiv Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda) \cdot q(n+1) > 0$. *If the experiment takes* $s(\lambda) = O(n^2(\rho(\lambda))^{-2} \log((nq \cdot \rho(\lambda))^{-1}))$ *samples when computing the estimate* $\tau'(VIEW_{\mathscr{A}})$, *then*

$$|Pr[\mathscr{G}_1] - (\frac{1}{2} + (Pr[\mathscr{G}_2] - \frac{1}{2}) \cdot 4q(n+1))| \leq \rho(\lambda).$$

**Game 3** This game is similar to Game 2 except that the public parameters and the trapdoors are generated differently. We suppose now that $Z$ is equal to $e(g_1, g_2)^{\theta\delta\omega}$.

**Setup.** $\mathscr{B}$ computes the public parameters *params* as follows.

First, $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes $g_2^{\beta}$. It then selects at random $\alpha \in_R \mathbb{Z}_p$ and generates $g_1^{\alpha^i}$ for $i \in [1,m] \cup [m+2, 2m]$ and $g_2^{\alpha^i}$ for $i \in [1,m]$. Futhermore, it picks at random $a_1, \cdots, a_m, c \in_R \mathbb{Z}_p$ and computes $g_1^{a_1}, \cdots, g_1^{a_m}, g_1^c$. In addition, the challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i \in [1,m] \cup [m+2,2m]}, \{g_2^{\alpha^i}\}_{i \in [1,m]}, g_2^{\beta}, \{g_1^{a_i}\}_{i \in [1,m]}, g_2^{\omega}, g_1^c, \mathsf{OTS})$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction.

The challenger also outputs the public description $(h_0, h_1, \cdots, h_n)$ of the hash function $H$ as defined in Game 2.

**Query Phase 1.** $\mathscr{A}$ can adaptively issue queries as follows.

- *First Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{a_i}, g_2^{\omega})^{cr_1 L_1^R}$ and $g_2^{r_1 a_i}$ for $i \in [1,m]$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R, i}$.

- *Update Key Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the update key generation oracle, then $\mathscr{B}$ randomly selects $\gamma, s_j, r_j$ in $\mathbb{Z}_p$ and generates $g_2^{s_j}$, $g_2^{r_j}$, $(g_1^{\gamma} \cdot \prod_{k \in S_j} g_1^{\alpha^{m+1-k}})^{s_j}$, for $S_j \subseteq [1,m]$, and $e(g_1^{\alpha}, g_2^{\alpha^m})^{s_j} \cdot e(g_1^c, g_2^{\omega})^{r_j L_j^R - r_1 L_1^R}$. The challenger forwards these elements to $\mathscr{A}$ as the update key $UK_{L_j^R}$.

- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{a_i}, g_2^{\omega})^{cr_j L_j^R}$ and $g_2^{a_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R, i}$.

- *Trapdoor Query* $< w_i^R, L_j^R >$. Suppose that $\mathscr{A}$ queries $(w_i^R, L_j^R)$ to the trapdoor generation oracle. If $y(w_i^R) \neq 0 \mod p$, then $\mathscr{B}$ selects $v, z$ at random in $\mathbb{Z}_p$, generates $e(g_1^{a_i}, g_2^{\omega})^{cr_j L_j^R} \cdot e(g_1^{\delta}, g_2^{\omega})^{a_i x(w_i^R)} \cdot Z^{a_i y(w_i^R)}$, $e(g_1, g_2^{\delta})^{a_i x(w_i^R)} \cdot e(g_1^{\theta}, g_2^{\delta})^{a_i y(w_i^R)}$, $g_1^{(\beta - w_i^R)v} \cdot g_1^{a_i z x(w_i^R)} \cdot (g_1^{\theta})^{a_i z y(w_i^R)}$, $(g_2^{\omega})^{a_i z}$, $g_2^{a_i r_j}$ and $g_1^v$ as the trapdoor $Trap_{w_i^R, L_j^R}$, and gives these elements to $\mathscr{A}$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a keyword pair $(w_0, w_1)$. The challenger first chooses a random bit $\mu \in_R \{0,1\}$ and lets the challenge keyword be $w^* = w_\mu$. It also defines a label $L^*$. It selects $v, z$ at random in $\mathbb{Z}_p$ and generates $T_1 = e(g_1^{a_i}, g_2^{\omega})^{cr_j L^*} \cdot e(g_1^{\delta}, g_2^{\omega})^{a_i x(w^*)} \cdot Z^{a_i y(w^*)}$, $T_2 = e(g_1, g_2^{\delta})^{a_i x(w^*)} \cdot e(g_1^{\theta}, g_2^{\delta})^{a_i y(w^*)}$, $T_3 = g_1^{(\beta - w^*)v} \cdot g_1^{a_i z x(w^*)} \cdot (g_1^{\theta})^{a_i z y(w^*)}$, $T_4 = (g_2^{\omega})^{a_i z}$, $T_5 = g_2^{a_i r_j}$ and $T_6 = g_1^v$ as the elements of the trapdoor $Trap^*$. The challenger sends the challenge trapdoor $Trap^*$ to the adversary.

When $Z = e(g_1, g_2)^{\theta\delta\omega}$, then $Trap^*$ is a valid challenge trapdoor to $\mathscr{A}$ as in the real attack. When $Z$ is random in $\mathbb{G}_T$, then $e(g_1^{a_i}, g_2^{\omega})^{cr_j L_j^R} \cdot e(g_1^{\delta}, g_2^{\omega})^{a_i x(w^*)} \cdot Z^{a_i y(w^*)}$ is a uniform element in $\mathbb{G}_T$, and thus the trapdoor gives no information about the challenger's bit $\mu$.

**Query Phase 2.** $\mathscr{A}$ continues to make queries as in the Query Phase 1. The restriction is that $< w_i^R, L_j^R >$ are not allowed to be queried as trapdoor queries if $< w_i^R, L_j^R >=< w_0, L^* >$ or $< w_i^R, L_j^R >=< w_1, L^* >$.

**Guess.** The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = e(g_1, g_2)^{\theta\delta\omega}$; otherwise, $\mathscr{B}$ outputs 0 meaning that $Z$ is a random element in $\mathbb{G}_T$.

We easily observe that the public parameters and the trapdoors are distributed identically in Game 2 and in Game 3. Therefore, we obtain that $Pr[\mathscr{G}_2] = Pr[\mathscr{G}_3]$.

**Game 4** Game 4 is similar to Game 3 except that the value $Z = e(g_1, g_2)^{\theta\delta\omega}$ is replaced by a random value $Z$ in $\mathbb{G}_T$, which is chosen at the beginning of the game. Therefore, $|Pr[\mathscr{G}_3] - Pr[\mathscr{G}_4]| \leq Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)$.

To explain why, we suppose that the tuple $(g_1, g_1^{\theta}, g_1^{\delta}, g_2, g_2^{\delta}, g_2^{\omega}, Z)$ is given to the challenger, such that $Z$ is either equal to $Z = e(g_1, g_2)^{\theta\delta\omega}$ or to a random element in $\mathbb{G}_T$. When $Z = e(g_1, g_2)^{\theta\delta\omega}$, we perfectly simulate the adversary in Game 3. When $Z \in_R \mathbb{G}_T$, then $e(g_1^{a_i}, g_2^{\omega})^{cr_j L_j^R} \cdot e(g_1^{\delta}, g_2^{\omega})^{a_i x(w^*)} \cdot Z^{a_i y(w^*)}$ is uniformly random in $\mathbb{G}_T$, and thus we perfectly simulate the adversary in Game 4. This means that if $\mathscr{A}$ can distinguish between Game 3 and Game 4, then the two possible values for $Z$ can be distinguished with the same probability. Therefore, when $Z$ is uniformly random in $\mathbb{G}_T$, we have that $Pr[\mathscr{G}_4] = \frac{1}{2}$.

**Analysis** We have described the simulations of all the games, and now, we can complete the proof by bounding the advantage of $\mathscr{A}$ in the IND-KGA security game as follows:

$$\begin{aligned}
Adv^{IND-KGA}_{\mathscr{A}}(\lambda) &= |Pr[\mathscr{G}_1] - \frac{1}{2}| \leq |(Pr[\mathscr{G}_2] - \frac{1}{2}) \cdot 4q(n+1)| + \rho(\lambda) \\
&\leq |(Pr[\mathscr{G}_3] - \frac{1}{2})| \cdot 4q(n+1) + \rho(\lambda) \\
&\leq (|Pr[\mathscr{G}_4] - \frac{1}{2}| + Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda)) \cdot 4q(n+1) + \rho(\lambda) \\
&= Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda) \cdot 4q(n+1) + \rho(\lambda)
\end{aligned}$$

where $\rho(\lambda) = Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda) \cdot q(n+1) > 0$. Thus, we obtain that $Adv^{IND-KGA}_{\mathscr{A}}(\lambda) \leq Adv^{DBDH}_{\mathscr{B},\mathbb{G}_T}(\lambda) \cdot 5q(n+1)$.

### 6.4 Collusion Resistance (CR)

**Theorem 4.** *The CBEKS scheme is collusion resistant without the random oracle model assuming that the m-DBDHE assumption holds.*

**Proof** Suppose that there exists a PPT adversary $\mathscr{A}$ that can attack the collusion resistance of the CBEKS scheme in the standard model with advantage $Adv^{CR}_{\mathscr{A}}(\lambda) \geq \varepsilon$. We build a challenger $\mathscr{B}$ that has advantage at least $\varepsilon$ in solving the $m$-DBDHE problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. $\mathscr{B}$ receives a random $m$-DBDHE problem instance $(g_1, g_1^{\delta}, g_1^{\alpha}, \cdots, g_1^{\alpha^m}, g_1^{\alpha^{m+2}}, \cdots, g_1^{\alpha^{2m}}, g_2, g_2^{\delta}, g_2^{\alpha}, \cdots, g_2^{\alpha^m})$ and $Z$ that is either equal to $e(g_1, g_2)^{\delta\alpha^{m+1}}$ or a random element in $\mathbb{G}_T$.

**Initialization.** $\mathscr{B}$ receives from $\mathscr{A}$ the group $S^*$ of receivers that the adversary wants to attack.

**Setup.** $\mathscr{B}$ computes the public parameters *params* and the secret keys $sk_{R,i}$ of the receivers in $[1,m] \setminus S^*$ as follows. First, $\mathscr{B}$ chooses at random $\beta \in_R \mathbb{Z}_p$ and computes $g_1^{\beta}, g_2^{\beta}$. It then selects at random $u \in \mathbb{Z}_p$ and generates $g_1^u \cdot (\prod_{k \in S^*} g_1^{\alpha^{m+1-k}})^{-1}$. Suppose that it sets this value equal to $g_1^{\gamma}$ for an unknown $\gamma$. It

also picks at random $a_1, \cdots, a_m, b, c \in_R \mathbb{Z}_p$ and computes $g_1^{a_1}, \cdots, g_1^{a_m}, g_2^b, g_1^c$. In addition, the challenger picks at random $A, B \in_R \mathbb{G}_2$ and chooses a strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

Finally, the challenger gives $\mathscr{A}$ the public parameters $params = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, A, B, \{g_1^{\alpha^i}\}_{i\in[1,m]\cup[m+2,2m]}, \{g_2^{\alpha^i}\}_{i\in[1,m]}, g_2^{\beta}, \{g_1^{a_i}\}_{i\in[1,m]}, g_2^b, g_1^c, \mathsf{OTS})$. Note that since the exponents in $\mathbb{Z}_p$ are uniformly chosen at random, these public parameters have an identical distribution to that in the actual construction.

In addition, $\mathscr{B}$ forwards $\mathscr{A}$ the secret keys of the receivers in $[1,m] \setminus S^*$. For all $i \notin S^*$, the challenger generates the secret key $sk_{R,i}$ as $(a_i, g_1^{\beta}, (g_1^{\alpha^i})^u \cdot (\prod_{k\in S^*} g_1^{\alpha^{m+1-k+i}})^{-1})$. Note that

$$(g_1^{\alpha^i})^u \cdot (\prod_{k\in S^*} g_1^{\alpha^{m+1-k+i}})^{-1} = (g_1^u \cdot (\prod_{k\in S^*} g_1^{\alpha^{m+1-k}})^{-1})^{\alpha^i} = (g_1^u \cdot (\prod_{k\in S^*} g_1^{\alpha^{m+1-k}})^{-1})^{\alpha^i} = (g_1^{\gamma})^{\alpha^i}$$

as required. Moreover, since $i \notin S^*$, the product defining the third element of $i$'s secret key does not include the element $g_1^{\alpha^{m+1}}$. We can so observe that $\mathscr{B}$ has all the necessary values to compute the secret keys $sk_{R,i}$ for $i \notin S^*$.

**Query Phase 1.** $\mathscr{A}$ makes the following queries:

- *First Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the first certificate query generation oracle, then $\mathscr{B}$ picks at random $r_1 \in_R \mathbb{Z}_p$ and computes $e(g_1^{a_i}, g_2^b)^{cr_1 L_1^R}$ and $g_2^{r_1 a_i}$ for $i \in S \subseteq S^*$. It sends these two elements to $\mathscr{A}$ as the first certificate $Cert_{L_1^R,i}$.

- *Refreshed Certificate Query* $< L_j^R >$. If $\mathscr{A}$ queries $L_j^R$ to the refreshed certificate generation oracle, then $\mathscr{B}$ picks at random $r_j \in_R \mathbb{Z}_p$ and sends the elements $e(g_1^{a_i}, g_2^b)^{cr_j L_j^R}$ and $g_2^{a_i r_j}$ to $\mathscr{A}$ as the refreshed certificate $Cert_{L_j^R,i}$ for $i \in S \subseteq S^*$.

**Challenge.** Once the adversary decides that the Query Phase 1 is over, it outputs a label pair $(L_0, L_1)$. The challenger answers by choosing a random bit $\mu \in_R \{0,1\}$ and by setting the challenge label $L^* = L_\mu$. $\mathscr{B}$ picks at random $r_{j-1}, r_j \in_R \mathbb{Z}_p$ and computes the challenge update key $UK^* = (g_2^{\delta}, g_2^{r_j}, (g_1^{\delta})^u, Z \cdot e(g_1^c, g_2^b)^{r_j L^* - r_{j-1}(L^*-1)})$. We denote by $L^* - 1$ the label preceding the label $L^*$.

Therefore, $UK^*$ is a valid update key to $\mathscr{A}$'s view, by writing $g_2^{\delta} = g_2^{s_j}$ for an unknown $s_j \in \mathbb{Z}_p$. Then, we get that $g_1^{\delta u} = (g_1^u \cdot (\prod_{k\in S} g_1^{\alpha^{m+1-k}})^{-1} \cdot (\prod_{k\in S} g_1^{\alpha^{m+1-k}}))^{\delta} = (g_1^u \cdot (\prod_{k\in S} g_1^{\alpha^{m+1-k}})^{-1} \cdot (\prod_{k\in S} g_1^{\alpha^{m+1-k}}))^{s_j}$.

When $Z = e(g_1, g_2)^{\delta\alpha^{m+1}}$, then $UK^*$ is a valid challenge update key to $\mathscr{A}$ as in the real attack. When $Z$ is random in $\mathbb{G}_T$, then $Z \cdot e(g_1^c, g_2^b)^{r_j L^* - r_{j-1}(L^*-1)}$ is a uniform element in $\mathbb{G}_T$, and thus the update key gives no information about the challenger's bit $\mu$.

**Query Phase 2.** $\mathscr{A}$ issues a number of queries as in the Query Phase 1. The restriction is that $< L_j^R >$ are not allowed to be queried as first certificate or refreshed certificate queries if $< L_j^R > = < L_0 >$ or $< L_j^R > = < L_1 >$.

**Guess.** The adversary outputs a bit $\mu' \in \{0,1\}$. If $\mu' = \mu$, then $\mathscr{B}$ outputs 1 meaning that $Z = e(g_1, g_2)^{\delta\alpha^{m+1}}$; otherwise, $\mathscr{B}$ outputs 0 meaning that $Z$ is a random element in $\mathbb{G}_T$.

**Analysis** When $Z = e(g_1, g_2)^{\delta\alpha^{m+1}}$, then the adversary must satisfy $|Pr[\mu' = \mu] - \frac{1}{2}| \geq \varepsilon$. When $Z \in_R \mathbb{G}_T$, then $Z \cdot e(g_1^c, g_2^b)^{r_j L^* - r_{j-1}(L^*-1)}$ is uniformly random in $\mathbb{G}_T$, and thus $Pr[\mu' = \mu] = \frac{1}{2}$. It follows that we have $Adv_{\mathscr{B},\mathbb{G}_T}^{m-DBDHE}(\lambda) \geq \varepsilon$.

# 7 Performance and Observations

Our CBEKS scheme remains efficient and practical since the size of the ciphertexts and the trapdoors is constant. In addition, the secret keys of the involved entities (namely, the receivers, the certifier and the server) have constant size. Although the public parameters have a size linear in the number $m$ of receivers, we observe that these elements are set up only once, at the beginning of the protocol. Therefore, this is not cumbersome for our scheme.

We evaluate the efficiency of our scheme CBEKS in Table 2. We use results of cryptographic operation implementations (exponentiations and pairings) using the MIRACL library, provided by Certivox for the MIRACL Authentication Server Project Wiki. All the following experiments are based on Borland C/C++ Compiler/Assembler and tested on a processor 2.4 GHz Intel i5 520M.

Since we consider asymmetric pairings, we choose a system based on AES with a 80-bit key and a Cocks-Pinch curve over $\mathbb{GF}_p$, for a 512-bit modulus $p$ and an embedding degree equal to 2. We assume that there are $m = 100$ receivers and there are $n = 48$ bits coding a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ with 6 characters, such as "urgent".

| | Exponentiation in $\mathbb{G}_1$ | Exponentiation in $\mathbb{G}_2$ | Exponentiation in $\mathbb{G}_T$ | Pairings |
|---|---|---|---|---|
| Time/computation | 0.51 | 0.51 | 0.12 | 1.14 |
| Setup | 205.2 | 52.02 | - | - |
| Encrypt | 25.5 | 2.55 | - | - |
| CertGen | 51 | 51 | | 114 |
| UpdtKeyGen | 1.02 | 1.02 | 0.36 | 3.42 |
| UpdtCert | - | - | 0.12 | 4.56 |
| TrapGen | 52.52 | 1.02 | | 2.28 |
| Test | - | 0.51 | 0.36 | 7.98 |

Table 2: Timings for the asymmetric pairing-based CBEKS scheme. Times are in milliseconds.

We note that the total time in the algorithm Setup is substantial; however this algorithm should be run only once to generate the public parameters and the static secret keys for all the receivers, certifiers and servers. The algorithm Encrypt does not require too many computations from the uploader: the time is mainly a consequence from the fact that the keyword's elements have to be hashed. Then, the algorithm CertGen is significant since we consider the generation of 100 first certificates; nevertheless, the same argument from Setup applies for CertGen, i.e. first certificates are generated only once. Both the algorithms UpdtKeyGen and UpdtCert are run fastly, meaning that the certifier and the receivers can easily create update keys and update the certificates, respectively. The algorithm TrapGen is relatively efficient, meaning that a receiver can conveniently request for a medical document. Finally, the total time for running the algorithm Test is mainly due to the cost of pairing computations, and remains quick.

We recall that we see a label as a reference to some information about the receivers' access rights. More precisely, a label is linked to a collection of rights under the form of a unique number in $\mathbb{Z}_p$. We have to ensure that two labels are distinct if they do not refer to the same access right collection.

We argue that the certifier and the server know the labels and agree on their use by securily exchanging information among them, while none of the receivers should be aware of the labels' contents. Nevertheless, defining unique labels could turn to be a bottleneck. A more effective solution will embed time periods instead of labels. Indeed, the certifier and the server will proceed by using the current time period: the former will generate the first certificates and update keys by taking as input the current time period, while the latter will check the keyword and the validity of a receiver's certificate by using the

ciphertext and the current time period. However, defining time periods does not seem straightforward at first sight. In fact, we have to ensure that a time period is unique given a starting date and an ending date and that the receivers are not able to modify their certificates themselves in order to make them valid. Moreover, a receiver's certificate can be valid for a longer time period than the one used during the server's test; hence, we have to find a way to make time periods matching as long as the time period used by the server is strictly included in the time period embedded into the receiver's certificate.

We also observe that we give power to the server by letting it know and use labels for verification processes. One might prefer instead to let the uploader decide the choices of both the keyword and the label (or the time period according to the above remarks). Hence, in this case, the server would have just to check that keywords and labels match without being aware of any pieces of information from these components.

# 8 Conclusion

In this paper, we introduced the new primitive called Certificate-Based Encryption with Keyword Search (CBEKS) in order to tackle the problem of authorizing receivers in a sensitive environment to let them access and retrieve private medical documents (such as EHRs) securely. We constructed a scheme and proved it secure in the standard model: we showed that our CBEKS scheme is computationally consistent, indistinguishable against chosen keyword and ciphertext attacks, indistinguishable against keyword-guessing attacks and collusion resistant.

# Acknowledgements

This work is partially supported by ARC Linkage Project LP12020052.

# References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Journal of Cryptology*, 21(3):350–391, March 2008.

[2] J. Baek, R. Safavi-Naini, and W. Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *Proc. of the 9th International Conference on Information Security (ISC'06), Samos Island, Greece*, volume 4176 of *Lecture Notes in Computer Science*, pages 217–232. Springer Berlin Heidelberg, August-September 2006.

[3] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In *Proc. of the International Conference on Computational Science and Its Applications (ICCSA'08), Perugia, Italy*, volume 5072 of *Lecture Notes in Computer Science*, pages 1249–1259. Springer Berlin Heidelberg, June-July 2008.

[4] D. Boneh and X. Boyen. Short signatures without random oracles. In *Proc. of the 2004 Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer Berlin Heidelberg, May 2004.

[5] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer Berlin Heidelberg, May 2004.

[6] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Proc. of the 25th Annual International Cryptology Conference on Advances in Cryptology*

*(CRYPTO'05), Santa Barbara, California, USA*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer Berlin Heidelberg, August 2005.

[7] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Proc. of the 3rd VLDB Workshop on Secure Data Management (SDM'06), Seoul, Korea*, volume 4165 of *Lecture Notes in Computer Science*, pages 75–83. Springer Berlin Heidelberg, September 2006.

[8] C. Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *Proc. of the 13th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'07), Kuching, Malaysia*, volume 4833 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin Heidelberg, December 2007.

[9] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Revised Papers of the 2003 ACM CCS Workshop on Digital Rights Management Workshop (DRM'02), Washington, DC, USA*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer Berlin Heidelberg, November 2003.

[10] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proc. of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'84), Santa Barbara, California, USA*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin Heidelberg, August 1984.

[11] L. Fang, W. Susilo, C. Ge, and J. Wang. A secure channel free public key encryption with keyword search scheme without random oracle. In *Proc. of the 8th International Conference on Cryptology and Network Security (CANS'09), Kanazawa, Japan*, volume 5888 of *Lecture Notes in Computer Science*, pages 248–258. Springer Berlin Heidelberg, December 2009.

[12] L. Fang, W. Susilo, C. Ge, and J. Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Science*, 238:221–241, 2013.

[13] A. Fiat and M. Naor. Broadcast encryption. In *Proc. of the 13th Annual International Cryptology Conference (CRYPTO'93), Santa Barbara, California, USA*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin Heidelberg, August 1993.

[14] T. Fuhr and P. Paillier. Decryptable searchable encryption. In *Proc. of the 1st International Conference on Provable Security (ProvSec'07), Wollongong, NSW, Australia*, volume 4784 of *Lecture Notes in Computer Science*, pages 228–236. Springer Berlin Heidelberg, November 2007.

[15] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'03), Warsaw, Poland, LNCS*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer Berlin Heidelberg, 2003.

[16] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Proc. of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'09), Cologne, Germany*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer Berlin Heidelberg, April 2009.

[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of the ACM Conference on Computer and Communications Security (CCS'06), Alexandria, Virginia, USA*, pages 89–98. ACM, October-November 2006.

[18] C. Gritti, W. Susilo, and T. Plantard. Efficient file sharing in electronic health records. In *Proc. of the 11th International Conference on Information Security Practice and Experience (ISPEC'15), Beijing, China*, volume 9065 of *Lecture Notes in Computer Science*, pages 499–513. Springer International Publishing, May 2015.

[19] C. Gu, Y. Zhu, and H. Pan. Efficient public key encryption with keyword search schemes from pairings. In *Revised Selected Papers of the 3rd SKLOIS Conference on Information Security and Cryptology (Inscrypt'07), Xining, China*, volume 4990 of *Lecture Notes in Computer Science*, pages 372–383. Springer-Verlag, August-September 2008.

[20] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee. Constructing peks schemes secure against keyword guessing attacks is possible? *Computer Communications*, 32(2):394–396, February 2009.

[21] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random

oracles. In *Proc. of the 11th Australasian Conference on Information Security and Privacy (ACISP'06), Melbourne, Australia*, volume 4058 of *Lecture Notes in Computer Science*, pages 336–347. Springer Berlin Heidelberg, July 2006.

[22] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proc. of the 21st Annual International Cryptology Conference on Advances in Cryptolog (CRYPTO'01), Santa Barbara, California, USA*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer Berlin Heidelberg, August 2001.

[23] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. of the 7th Conference on USENIX Security Symposium (SSYM'98), San Antonio, Texas, USA*, volume 7, pages 217—-228. USENIX Association, January 1998.

[24] D.-H. Phan, D. Pointcheval, S. F. Shahandashti, and M. Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. In *Proc. of the 17th Australasian on Information Security and Privacy (ACISP'12), Wollongong, NSW, Australia*, volume 7372 of *Lecture Notes in Computer Science*, pages 308–321. Springer Berlin Heidelberg, July 2012.

[25] D. H. Phan, D. Pointcheval, and M. Strefler. Security notions for broadcast encryption. In *Proc. of the 9th International Conference on Applied Cryptography and Network Security (ACNS'11), Nerja, Spain*, volume 6715 of *Lecture Notes in Computer Science*, pages 377–394. Springer Berlin Heidelberg, June 2011.

[26] H. S. Rhee, J. H. Park, and D. H. Lee. Generic construction of designated tester public-key encryption with keyword search. *Infornation Science*, 205:93–109, November 2012.

[27] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Improved searchable public key encryption with designated tester. In *Proc. of the 4th International Symposium on Information, Computer and Communications Security (ASIACCS'09), Sydney, NSW, Australia*, pages 376–379. ACM, March 2009.

[28] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771, May 2010.

[29] B. Waters. Efficient identity-based encryption without random oracles. In *Proc. of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'05), Aarhus, Denmark*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer Berlin Heidelberg, May 2005.

[30] B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *Proc. of the 29th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'09), Santa Barbara, California, USA*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer Berlin Heidelberg, August 2009.

[31] W.-C. Yau, R. C. W. Phan, S.-H. Heng, and B.-M. Goi. Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. *International Journal of Computer Mathematics*, 90(12):2581–2587, December 2013.

## Author Biography

**Clémentine Gritti** received the BS Degree in pure mathematics in 2010 and the MS Degree in computer science from Joseph Fourier University, Grenoble, France in 2012. She is currently a PhD candidate at the University of Wollongong, Australia. Her current research interests include cryptography and information security, and in particular public-key cryptographic primitives and provable security.

**Willy Susilo** received the Ph.D. degree in computer science from the University of Wollongong, Wollongong, Australia. He is a Professor and the Head of School of Computing and Information Technology. He is also the Director of Centre for Computer and Information Security Research, University of Wollongong. He has been awarded the prestigious ARC Future Fellow awarded by the Australian Research Council. His main research interests include cryptography and information security. His main contribution is in the area of digital signature schemes and encryption schemes. He has served as a program committee member in dozens of international conferences.

**Thomas Plantard** received the MS and Ph.D. Degrees in computer science from the Universite de Bordeaux in 2002 and the Universite Montpellier 2, France, in 2005, respectively. Since September 2006, he has a postdoctoral position at the University of Wollongong, Australia. His research interests include cryptography and lattice theory.