A preliminary version of this paper appears in *Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2008*, ACM Press, 2008. This is the full version.

Identity-based Encryption with Efficient Revocation

Alexandra Boldyreva* Vipul Goyal† Virendra Kumar‡

Abstract

Identity-based encryption (IBE) is an exciting alternative to public-key encryption, as IBE eliminates the need for a Public Key Infrastructure (PKI). Any setting, PKI- or identity-based, must provide a means to revoke users from the system. Efficient revocation is a well-studied problem in the traditional PKI setting. However in the setting of IBE, there has been little work on studying the revocation mechanisms. The most practical solution requires the senders to also use time periods when encrypting, and all the receivers (regardless of whether their keys have been compromised or not) to update their private keys regularly by contacting the trusted authority. We note that this solution does not scale well – as the number of users increases, the work on key updates becomes a bottleneck. We propose an IBE scheme that significantly improves key-update efficiency on the side of the trusted party (from linear to logarithmic in the number of users), while staying efficient for the users. Our scheme builds on the ideas of the Fuzzy IBE primitive and binary tree data structure, and is provably secure.

Keywords: Identity-based encryption, revocation, provable security.

*School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA. E-mail: sasha@gatech.edu. Supported in part by NSF CAREER award 0545659 and NSF Cyber Trust grant 0831184.

†Dept. of Computer Science, University of California, Los Angeles, CA, USA. E-mail: vipul@cs.ucla.edu. The author was supported in part from grants from the NSF ITR and Cybertrust programs (including grants 0627781, 0456717, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, a Microsoft Research Fellowship, an Alfred P. Sloan Foundation Fellowship, and an Okawa Foundation Research Grant.

‡School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA. E-mail: virendra@gatech.edu. Supported in part by the grant of the first author.

Contents

1 Introduction

1.1 Motivation

Identity (ID)-based encryption, or IBE for short, is an exciting alternative to public-key encryption, which eliminates the need for a Public Key Infrastructure (PKI) that makes publicly available the mapping between identities, public keys, and validity of the latter. The senders using an IBE do not need to look up the public keys and the corresponding certificates of the receivers, because the identities (e.g. emails or IP addresses) together with common public parameters are sufficient for encryption. The private keys of the users are issued by a trusted third party called the private key generator (PKG). Ideas of identity-based cryptography go back to 1984 and Shamir [24], but the first IBE scheme was constructed by Boneh and Franklin only in 2001 [6], building on the progress in elliptic curves with bilinear pairings.

Any setting, PKI- or identity-based, must provide a means to revoke users from the system, e.g. if their private keys get compromised. In a PKI setting a certification authority informs the senders about expired or revoked keys of the users via publicly available digital certificates and certificate revocation lists.

As a solution to this problem for IBE, Boneh and Franklin [6] suggested that users renew their private keys periodically, e.g. every week, and senders use the receivers' identities concatenated with the current time period, e.g. "week 15 of 2008". Notice that since only the PKG's public key and the receiver's identity are needed to encrypt, and there is no way to communicate to the senders that an identity has been revoked, such a mechanism to regularly update users' private keys seems to be the only viable solution to the revocation problem. This means that all users, regardless of whether their keys have been exposed or not, have to regularly get in contact with the PKG, prove their identity and get new private keys. The PKG must be online for all such transactions, and a secure channel must be established between the PKG and each user to transmit the private key. Taking scalability of IBE deployment into account, we observe that for a very large number of users this may become a bottleneck.

We note that alternatively, in order to avoid the need for interaction and a secure channel, the PKG may encrypt the new keys of non-revoked users under their identities and the previous time period, and send the ciphertexts to these users (or post them online). With this approach, for every non-revoked user in the system, the PKG is required to perform one key generation and one encryption operation per key update. We note that this solution, just as the original suggestion, requires the PKG to do work linear in the number of users, and does not scale well as the number of users grow. The goal of the paper is to study this problem and find solutions to alleviate it.

1.2 Related work

Efficient revocation is a well-studied problem in the traditional PKI setting, e.g. [18, 21, 1, 20, 19, 11, 12]. However in the setting of IBE, there has been little work on studying the revocation mechanisms. Hanaoka et al. [15] propose a way for the users to periodically renew their private keys without interacting with the PKG. The PKG publicly posts the key update information, which is much more convenient. However, each user needs to posses a tamper-resistant hardware device. This assumption makes the solution rather cumbersome.

Revocation has been studied in the ID-based setting with mediators [5, 17]. In this setting there is a special semi-trusted third party called a mediator who holds shares of all users' private keys and helps users to decrypt each ciphertext. If an identity is revoked then the mediator is instructed

to stop helping the user. But we want to focus on a much more practical standard IBE setting where users are able to decrypt on their own.

The goal of broadcast encryption is to prevent revoked users from accessing secret information being broadcast. The broadcast encryption solutions, however, and in particular ID-based broadcast encryption ones, do not directly translate into solutions for our problem. In broadcast encryption, a non-revoked user can help a revoked user gain access to the sensitive information being broadcast (since this information is the same for all parties). On the other hand, in the IBE setting a revoked user, or the adversary holding its private key, should not be able to decrypt messages even if it colludes with any number of non-revoked users.

Thus, to the best of our knowledge, the solution proposed by Boneh and Franklin in [6] remains the most practical user revocation solution in the IBE setting.

1.3 Contributions

We propose a new way to mitigate the limitation of IBE with regard to revocation, and improve efficiency of the previous solution. We want to remove interaction form the process of key update, as keeping the PKG online can be a bottleneck, especially if the number of users is very large. At the same time we do not want to employ trusted hardware and we want to significantly minimize the work done by the PKG and users.

First we define the Revocable IBE primitive and its security model that formalizes the possible threats. The model, of course, takes into account all adversarial capabilities of the standard IBE security notion. I.e. the adversary should be able to learn the private keys of the users with identities of its choice, and in the case of chosen-ciphertext attack to also see decryptions under the private key of the challenge identity of the ciphertexts of its choice. The adversary should not be able to learn any partial information about the messages encrypted for the challenge identity. In addition we consider the adversary having access to periodic key updates (as we assume this information is public) and being able to revoke users with IDs of its choice. The adversary should not be able to learn any partial information about the messages encrypted for any revoked identity when the ciphertext is created after the time of revocation (i.e. for the ID containing the time past the revocation time).

We show that it is possible to reduce the amount of work a PKG has to do for key updates and the total size of key updates to *logarithmic* in the number of users, while keeping the key update process non-interactive, and encryption and decryption efficient.

Our idea is to build on the Fuzzy IBE construction by Sahai and Waters [23]. The Fuzzy IBE primitive provides some sort of error-tolerance, i.e. identities are viewed as sets of attributes, and a user can decrypt if it possesses keys for enough of (but not necessarily all) attributes a ciphertext is encrypted under. At the same time, colluding users cannot combine their keys to decrypt a ciphertext which none of them were able to decrypt independently.

We propose to combine the Fuzzy IBE construction from [23] with the binary tree data structure, which was previously used to improve efficiency of revocations in the PKI setting [21, 1]. In order to decrypt a ciphertext encrypted for an identity and time period the user must possess the keys for these two attributes. The PKG publicly posts and regularly updates the keys for the current time attribute. Even though the time attributes are the same for all users, this does not have to compromise security, thanks to the collusion-resistance property of Fuzzy IBE. To reduce the size of key updates from linear to logarithmic in the number of users, the binary tree data structure is used. Here we employ a trick to modify the Fuzzy IBE scheme in such a way that collusion of some users (corresponding to non-revoked users in our scheme) on some attributes (i.e. time attribute)

is possible. We provide more details and present the full construction in Section 4.

While our scheme provides major computation and bandwidth efficiency improvements at the stage of key update, it also permits efficient encryption and decryption. We show that our scheme *provably* guarantees security assuming the decisional bilinear Diffie-Hellman (DBDH) problem is hard, which is a quite common assumption nowadays (cf. e.g. [3, 23, 25, 14]).

We also show two ways to address chosen-ciphertext attack. Our first solution is to modify our scheme by additionally employing a strongly-unforgeable one-time signature scheme in a manner somewhat similar to that from [7, 14]. We also show that it is possible to employ the Fujisaki-Okamoto (FO) transform [9, 10]. Security of the latter solution relies on the random oracle model [2], but unlike the former solution, it is generic, in that it can be applied to any Revocable IBE scheme.

Since the existing Fuzzy IBE schemes are only secure in the weaker selective-ID model [8], where the adversary has to declare the challenge identity up front, with the above approach we can only achieve selective-ID security as well. We leave it as an interesting open problem to achieve full security without such limitation.

Finally we note that the problem of revocation is equally important for Fuzzy IBE and attribute-based encryption (ABE) [14] schemes. While the same periodic key update solution due to Boneh and Franklin applies, it similarly limits scalability. We show that it is possible to extend our techniques to provide efficient non-interactive key update to Fuzzy IBE and ABE schemes.

2 Notation and Conventions

Let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{R}$ denote the set of real numbers. For $n \in \mathbb{N}$, let $\mathbb{Z}_n$ denote the set of integers modulo n and $\mathbb{Z}_n^*$ denote the set $\mathbb{Z}_n \setminus 0$. We denote by $\{0,1\}^*$ the set of all binary strings of finite length. If x is string then $|x|$ denotes its length in bits and if $x \in \mathbb{R}$ then $|x|$ denotes its absolute value. If x, y are strings then $x\|y$ denotes the concatenation of x and y, and we assume that x and y can be efficiently and unambiguously recovered from $x\|y$. If $\kappa \in \mathbb{N}$ then 1^κ denotes the string consisting of κ consecutive "1" bits. We denote by ϕ the empty set. If x, y are strings then $x\|y$ denotes the concatenation of x and y, and we assume that x and y can be efficiently and unambiguously recovered from $x\|y$. If S is a finite set then $s \xleftarrow{\$} \mathsf{S}$ denotes that s is selected uniformly at random from S . We will often write $s_1, s_2, \ldots, s_n \xleftarrow{\$} \mathsf{S}$ as a shorthand for $s_1 \xleftarrow{\$} \mathsf{S}; s_2 \xleftarrow{\$} \mathsf{S}; \ldots; s_n \xleftarrow{\$} \mathsf{S}$. When describing algorithms, $a \leftarrow b$ denotes that a is assigned the value b. If A is a randomized algorithm and $n \in \mathbb{N}$, then $a \xleftarrow{\$} A(i_1, i_2, \ldots, i_n)$ denotes that a is assigned the outcome of the experiment of running A on inputs $i_1, i_2, \ldots, i_n$. If A is deterministic, then we drop the dollar sign above the arrow. If $\mathsf{S} = \{s_1, s_2, \ldots, s_n\}$, then $\{x_s\}_{s \in \mathsf{S}}$ denotes the set $\{x_{s_1}, x_{s_2}, \ldots, x_{s_n}\}$. An adversary is an algorithm. By convention, the running time of an adversary includes that of its overlying experiment. All algorithms are assumed to be randomized and efficient (i.e. polynomial in the size of the input), unless noted otherwise. In the rest of the paper $\kappa \in \mathbb{N}$ is the security parameter, $n(\cdot)$ denotes a polynomial in κ, but for simplicity we use the notation n.

3 Revocable IBE and its Security

3.1 Syntax of Revocable IBE

We start with defining the general syntax of a Revocable IBE scheme.

Definition 3.1 [Revocable IBE] An *identity-based encryption with efficient revocation* or simply *Revocable IBE* scheme $\mathcal{RIBE} = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ is defined by seven algorithms and has associated message space $\mathcal{M}$, identity space $\mathcal{I}$ and time space $\mathcal{T}$. We assume that the size of $\mathcal{T}$ is polynomial in the security parameter. Each algorithm is run by either one of three types of parties - key authority, sender or receiver. Key authority maintains a revocation list rl and state st. Revocation list rl can be part of state st, but we keep it explicit for clarity. In what follows, we call an algorithm stateful only if it updates rl or st. We treat time as discrete as opposed to continuous.

- The stateful *setup* algorithm $\mathcal{S}$ (run by key authority) takes input security parameter 1^κ and number of users n, and outputs public parameters pk, master key mk, revocation list rl (initially empty) and state st.
- The stateful *private key generation* algorithm $\mathcal{SK}$ (run by key authority) takes input public parameters pk, master key mk, identity $\omega \in \mathcal{I}$ and state st, and outputs private key sk_ω and an updated state st.
- The *key update generation algorithm* $\mathcal{KU}$ (run by key authority) takes input public parameters pk, master key mk, key update time $t \in \mathcal{T}$, revocation list rl and state st, and outputs key update ku_t.
- The deterministic *decryption key generation* algorithm $\mathcal{DK}$ (run by receiver) takes input private key sk_ω and key update ku_t, and outputs decryption key $dk_{\omega,t}$ or a special symbol $\perp$ indicating that ω was revoked. (We say an identity ω was revoked at time t if revocation algorithm $\mathcal{R}$ was run by key authority on input (ω, t, rl, st) for any rl, st.)
- The *encryption* algorithm $\mathcal{E}$ (run by sender) takes input public parameters pk, identity $\omega \in \mathcal{I}$, encryption time $t \in \mathcal{T}$ and message $m \in \mathcal{M}$, and outputs ciphertext c. For simplicity and wlog we assume that ω, t are efficiently computable from c.
- The deterministic *decryption* algorithm $\mathcal{D}$ (run by receiver) takes input decryption key $dk_{\omega,t}$ and ciphertext c, and outputs a message $m \in \mathcal{M}$ or, a special symbol $\perp$ indicating that the ciphertext is invalid.
- The stateful *revocation* algorithm $\mathcal{R}$ (run by key authority) takes input identity to be revoked $\omega \in \mathcal{I}$, revocation time $t \in \mathcal{T}$, revocation list rl and state st, and outputs an updated revocation list rl.

The consistency condition requires that for all $\kappa \in \mathbb{N}$ and polynomials (in κ) n, all pk and mk output by setup algorithm $\mathcal{S}$, all $m \in \mathcal{M}, \omega \in \mathcal{I}, t \in \mathcal{T}$ and all possible valid states st and revocation lists[1] rl, if identity ω was not revoked before or, at time t then the following experiment returns 1 with probability 1:

$$(sk_\omega, st) \xleftarrow{\$} \mathcal{SK}(pk, mk, \omega, st)\ ;\ ku_t \xleftarrow{\$} \mathcal{KU}(pk, mk, t, rl, st)$$
$$dk_{\omega,t} \leftarrow \mathcal{DK}(sk_\omega, ku_t)\ ;\ c \xleftarrow{\$} \mathcal{E}(pk, \omega, t, m)$$
If $\mathcal{D}(dk_{\omega,t}, c) = m$ then return 1 else return 0. ∎

REMARKS. Note that we differentiate between the terms "private key" and "decryption key".

One can also define the decryption key generation algorithm that instead of private key sk_ω takes input the decryption key for the previous time period $dk_{\omega,t-1}$. We do not further discuss this version here since it is not used in our construction.

[1] A valid state is the one that is output by either setup algorithm $\mathcal{S}$ or private key generation algorithm $\mathcal{SK}$. A valid revocation list is the one that is output by either setup algorithm $\mathcal{S}$ or revocation algorithm $\mathcal{R}$.

3.2 Security of Revocable IBE

We define the *selective-revocable-ID* security for Revocable IBE schemes. Our security model captures the standard notion of selective-ID security but it also takes into account possible revocations. Since we explicitly consider time periods, in the beginning of the experiment in addition to the challenge identity the adversary also declares the challenge time. Just as in the standard slective-ID security definition the adversary can request to learn users' keys. In addition we let the adversary to revoke users of its choice (including the challenge identity) at any period of time and see all key updates. Unlike in the standard security model, we allow the adversary to learn the private key for the challenge identity, but only if it was revoked prior to or at the challenge time. The adversary is given a ciphertext of one of the two messages of its choice encrypted for the challenge identity and time. It has to guess which of the messages was encrypted.

First we define (selective) security against chosen-plaintext attack and then show how to extend the definition to chosen-ciphertext attack.

Definition 3.2 [sRID Security] Let $\mathcal{RIBE} = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ be a Revocable IBE scheme. The adversary first outputs the challenge identity and time, and also some information *state* it wants to preserve. Later it is given access to three oracles that correspond to the algorithms of the scheme. The oracles share state.[2] Since we use the simplified notation for the oracles, we define them now:

- The *private key generation* oracle $\mathcal{SK}(\cdot)$ takes input identity ω and runs $\mathcal{SK}(pk, mk, \omega, st)$ to return private key sk_ω.
- The *key update generation* oracle $\mathcal{KU}(\cdot)$ takes input time t and runs $\mathcal{KU}(pk, mk, t, rl, st)$ to return key update ku_t.
- The *revocation* oracle $\mathcal{R}(\cdot, \cdot)$ takes input identity ω and time t and runs $\mathcal{R}(\omega, t, rl, st)$ to update rl.

For adversary A and number of users n define the following experiments:

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},A,n}(1^\kappa) \\
\quad b \xleftarrow{\$} \{0,1\} \\
\quad (\omega^*, t^*, state) \xleftarrow{\$} A(1^\kappa) \\
\quad (pk, mk, rl, st) \xleftarrow{\$} \mathcal{S}(1^\kappa, n) \\
\quad (m_0, m_1, state) \xleftarrow{\$} A^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot)}(pk, state) \\
\quad c^* \xleftarrow{\$} \mathcal{E}(pk, \omega^*, t^*, m_b) \\
\quad d \xleftarrow{\$} A^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot)}(pk, c^*, state) \\
\quad \text{If } b = d \text{ return 1 else return 0.}
\end{array}
$$

The following conditions must always hold:

1. $m_0, m_1 \in \mathcal{M}$ and $|m_0| = |m_1|$.
2. $\mathcal{KU}(\cdot)$ and $\mathcal{R}(\cdot, \cdot)$ can be queried on time which is greater than or equal to the time of all previous queries i.e. the adversary is allowed to query only in non-decreasing order of time[3].

[2] To be more formal we could define a single oracle that maintains the state and invokes these oracles as subroutines. We do not do it for simplicity.

[3] This is wlog because, the adversary can query the oracles for all possible time periods, one by one.

Also, the oracle $\mathcal{R}(\cdot,\cdot)$ cannot be queried on time t if $\mathcal{KU}(\cdot)$ was queried on t.[4]

3. If $\mathcal{SK}(\cdot)$ was queried on identity ω^* then $\mathcal{R}(\cdot,\cdot)$ must be queried on (ω^*, t) for any $t \leq t^*$.

We define the *advantage* of the adversary $\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},A,n}(\kappa)$ as

$$2 \cdot \Pr\left[\, \mathbf{Exp}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},A,n}(1^\kappa) = 1 \,\right] - 1 \,.$$

The scheme $\mathcal{RIBE}$ is said to be *sRID-CPA secure* if the function $\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},A,n}(\cdot)$ is negligible in κ for any efficient A and polynomial n. ▌

Chosen-ciphertext attack. We extend the above definition in the standard way to take into account chosen-ciphertext attack. Whenever the adversary is given the oracles, it is also given the *decryption oracle* $\mathcal{D}(\cdot)$ that takes input ciphertext c and runs $\mathcal{D}(dk_{\omega^*,t}, c)$ to return message m or $\perp$. The usual restriction is that $\mathcal{D}(\cdot)$ cannot be queried on challenge ciphertext c^*. The advantage of the adversary $\mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{RIBE},A,n}(\kappa)$ and *sRID-CCA security* are defined analogously to the CPA setting.

4 Main Construction

Intuition. At a high level we build on the (large universe) construction of Fuzzy IBE [23] and the binary tree data structure. We briefly recall the Fuzzy IBE primitive ideas and the basics of the construction.

In the Fuzzy IBE construction from [23], users' keys and ciphertexts are associated with sets of descriptive attributes. A user's key can decrypt a particular ciphertext only if some number of attributes (so called "error-tolerance") match between the ciphertext and the key. The number of attributes used to encrypt and the error-tolerance are fixed during the setup. Security of Fuzzy IBE requires that different users should not be able to pool their attributes together in order to decrypt a ciphertext which none of them were able to decrypt individually. To prevent collusions, the key generation algorithm of Fuzzy IBE generates a random polynomial (of degree one less than the error-tolerance) for each user. This polynomial is used to compute keys corresponding to a set of attributes. Since all the keys are computed on different polynomials, they cannot be combined in any meaningful way.

In our IBE scheme messages are encrypted for two "attributes": identity of the receiver and time period. The decryption key is also computed for attributes identity and time, on a first-degree polynomial, meaning both attributes of the decryption key must match with those of a ciphertext in order to decrypt. We split the decryption key in two components corresponding to identity and time that we call private key and key update respectively. The private key is issued to each user by the key authority,[5] just like regular private keys in IBE. The key update is published by the key authority and is publicly available to all users. To be able to decrypt a ciphertext a user needs both the private key and the key update. Thus, when the key authority needs to revoke a user it may simply stop publishing key updates for that user. As we recalled above, in Fuzzy IBE the polynomial of a decryption key is selected at random to prevent collusion between different keys. Using Fuzzy IBE in a naive way would thus require computing key updates for each user separately. We use a different approach to reduce the number of key updates that key authority

[4] This is because we assume that the key update is done at the end of the time period t.

[5] We use a different name than PKG to emphasize a new way to handle revocations.

needs to compute. We use a binary tree of height h (with at least as many leaves as the number of users in the system) and assign a random polynomial to each node of the tree. Next, we associate each user to a unique leaf node. Every user gets keys (corresponding to its identity) computed on polynomials of all nodes on the path from the leaf node corresponding to that user to the root node. To be able to decrypt a ciphertext encrypted with time t, any user just needs one key update (corresponding to t) computed on any one of the polynomials of nodes on the path from the leaf node of the user to the root node. Thus, when no user is revoked, key authority just needs to publish the key update computed on the polynomial of the root node. When a subset of the users is revoked, key authority first finds the minimal set of nodes in the tree which contains an ancestor (or, the node itself) among all the leaf nodes corresponding to non-revoked users. Then, key authority publishes key updates on polynomials of the nodes in this set. We first address chosen-plaintext attack only, and later show how to extend the scheme to resist chosen-ciphertext attack as well.

Before we give a formal description of the scheme, we define bilinear maps (aka. pairings).

BILINEAR MAPS AND GROUP GENERATOR. Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p (so they are cyclic). A *pairing* is an efficiently computable map $\mathbf{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that the following two conditions hold:

- Bilinearity: For all $g_1, g_2 \in \mathbb{G}$ and $x, y \in \mathbb{Z}$, we have $\mathbf{e}(g_1^x, g_2^y) = \mathbf{e}(g_1, g_2)^{xy}$.
- Non-degeneracy: For any generator g of $\mathbb{G}$, $\mathbf{e}(g, g)$ is a generator of $\mathbb{G}_T$.

Note that $\mathbf{e}(\cdot, \cdot)$ is symmetric since $\mathbf{e}(g^x, g^y) = \mathbf{e}(g, g)^{xy} = \mathbf{e}(g^y, g^x)$.

A *bilinear group generator* $\mathcal{G}$ is an algorithm that on input 1^κ returns $\tilde{\mathbb{G}}$, which is a description of groups $\mathbb{G}, \mathbb{G}_T$ of order p and the bilinear map $\mathbf{e}$ as defined above, and also p and a generator g of $\mathbb{G}$. There can be numerous such prime order bilinear group generators. We will not specify a particular one but will use it as a parameter to the hardness assumption that we use for our security proof. The description of a group should specify the algorithms for group operations (multiplication, inverse and pairing), the algorithm for testing group membership, and also the random group element sampling algorithm. Here and further in the paper we assume that the group elements are uniquely encoded as strings.

CONSTRUCTION. We now specify the scheme $\mathcal{RIBE}[\mathcal{G}] = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ in detail. We assume that all users agree on how time is divided by time periods and how each time period is specified, e.g. by days and "04.14.08". In our $\mathcal{RIBE}$ scheme messages are encrypted using identity and time. Identity is a string associated with any user, e.g. an email "abc@xyz.com". Time indicates when the ciphertext is supposed to be decrypted, e.g. on 04.14.08. The message space $\mathcal{M}$ is $\mathbb{G}_T$. The identity space $\mathcal{I}$ is $\{0,1\}^*$, and the time space $\mathcal{T}$ is an arbitrary bitstring set of size polynomial in the security parameter. We require that the strings specifying identities and times can be distinguished, e.g. by reserving the most significant bit (MSB) 0 for identity strings and 1 for time strings. In our construction the identity and time strings are mapped to unique elements of $\mathbb{Z}_p^*$ (if needed, a collision-resistant hash function $\{0,1\}^* \to \mathbb{Z}_p^*$ can be used). From now on for simplicity we assume that identity and time are distinguished elements in $\mathbb{Z}_p^*$.

For $x, i \in \mathbb{Z}$, set $\mathsf{J} \subseteq \mathbb{Z}$ the *Lagrange coefficient* $\Delta_{i,\mathsf{J}}(x)$ is defined as

$$\Delta_{i,\mathsf{J}}(x) \stackrel{\text{def}}{=} \prod_{j \in \mathsf{J},\, j \neq i} \left(\frac{x - j}{i - j} \right).$$

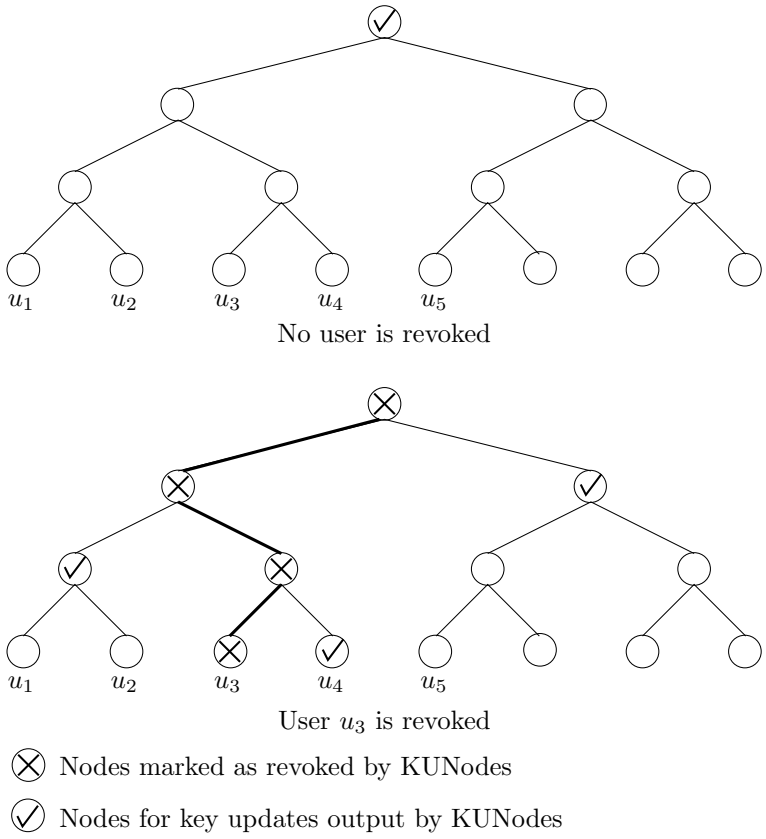

Figure 1: A pictorial description of the actions of KUNodes function used in Construction 4.1.

For $x \in \mathbb{Z}, g \in \mathbb{G}_T, \mathsf{J} \subseteq \mathbb{Z}, h_1, \ldots, h_{|\mathsf{J}|} \in \mathbb{G}$, we define

$$H_{g,\mathsf{J},h_1,\ldots,h_{|\mathsf{J}|}}(x) \stackrel{\text{def}}{=} g^{x^2} \prod_{i=1}^{|\mathsf{J}|} \left(h_i^{\Delta_{i,\mathsf{J}}(x)}\right) .$$

Our construction uses the binary tree data structure, so we introduce some notation here. We denote by root the root node. If v is a leaf node then $\mathsf{Path}(v)$ denotes the set of nodes on the path from v to root (both v and root inclusive). If v is a non-leaf node then v_l, v_r denote left and right child of v. We assume that nodes in the tree are uniquely encoded as strings, and the tree is defined by all of its nodes descriptions.

We also define a function KUNodes that is used to compute the minimal set of nodes for which key update needs to be published so that only non-revoked users at time t are able to decrypt ciphertexts.[6] The function takes input a binary tree T, revocation list rl and time t and outputs a set of nodes, which is the minimal set of nodes in T such that none of the nodes in rl with corresponding time $\leq t$ (users revoked on or before t) have any ancestor (or, themselves) in the set, and all other leaf nodes (corresponding to non-revoked users) have exactly one ancestor (or, themselves) in the set. The function operates as follows. First mark all the ancestors of revoked nodes as revoked, then output all the non-revoked children of revoked nodes. Refer to Figure 1 for

[6] A similar function was used in [1].

a pictorial depiction. Here is a formal specification.

$\mathsf{KUNodes}(\mathsf{T}, rl, t)$
  $\mathsf{X}, \mathsf{Y} \leftarrow \phi$
  $\forall (v_i, t_i) \in rl$
    if $t_i \leq t$ then add $\mathsf{Path}(v_i)$ to X
  $\forall x \in \mathsf{X}$
    if $x_l \notin \mathsf{X}$ then add x_l to Y
    if $x_r \notin \mathsf{X}$ then add x_r to Y
  If $\mathsf{Y} = \phi$ then add root to Y
  Return Y

We are now ready to present the description of Revocable IBE. We could not use the algorithms of the Fuzzy IBE construction from [23] in a black-box manner. The reason is that there the polynomial for each key is picked independently by the key generation algorithm. And in our construction some polynomials need to be shared by different keys. After we provide the details for each algorithm, we give some intuition and relation to the construction from [23] following "//" sign.

Construction 4.1 Let $\mathcal{G}$ be a prime order bilinear group generator. Let J be $\{1, 2, 3\}$.

- **Setup** $\mathcal{S}(1^\kappa, n)$:
  $(\tilde{\mathbb{G}}, p, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$; $a \xleftarrow{\$} \mathbb{Z}_p$; $g_1 \leftarrow g^a$; $g_2, h_1, h_2, h_3 \xleftarrow{\$} \mathbb{G}$. Let rl be an empty set and T be a binary tree with at least n leaf nodes.
  Return $pk = (g, g_1, g_2, h_1, h_2, h_3)$, $mk = a$; rl , $st = \mathsf{T}$.

  // Besides the additional outputs of rl and st, it is essentially the same as Setup of Fuzzy IBE where 2 out of 2 attributes need to be matched.

- **Private Key Generation** $\mathcal{SK}(pk, mk, \omega, st)$:
  Parse pk as $(g, g_1, g_2, h_1, h_2, h_3)$, mk as a, st as T.[7] Pick an unassigned leaf node v from T and store ω in that node.

  $\forall x \in \mathsf{Path}(v)$
    if a_x is undefined, then $a_x \xleftarrow{\$} \mathbb{Z}_p$,
    store a_x in node x,
    $r_x \xleftarrow{\$} \mathbb{Z}_p$; $D_x \leftarrow g_2^{a_x \omega + a} H_{g_2, \mathsf{J}, h_1, h_2, h_3}(\omega)^{r_x}$;
    $d_x \leftarrow g^{r_x}$.
  Return $sk_\omega = \{(x, D_x, d_x)\}_{x \in \mathsf{Path}(v)}$, st.

  // We note that a_x above fixes first-degree polynomial $q_x(y) = a_x y + a$ corresponding to node x. The algorithm computes the ω-components of the decryption key using the polynomials of all the nodes on the path from leaf node corresponding to ω to the root node.

[7] Every node x in T stores an element $a_x \in \mathbb{Z}_p$ and in addition, every leaf node stores an identity ω. If no such identity is stored at a leaf node we say that the leaf node is unassigned.

- **Key Update Generation** $\mathcal{KU}(pk, mk, t, rl, st)$:
  Parse pk as $(g, g_1, g_2, h_1, h_2, h_3)$, mk as a, st as T.

  $\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)$
  $r_x \xleftarrow{\$} \mathbb{Z}_p$; $E_x \leftarrow g_2^{a_x t + a} H_{g_2,\mathsf{J},h_1,h_2,h_3}(t)^{r_x}$;
  $e_x \leftarrow g^{r_x}$.
  Return $ku_t = \{(x, E_x, e_x)\}_{x \in \mathsf{KUNodes}(\mathsf{T},rl,t)}$.

  // The algorithm first finds a minimal set of nodes which contains an ancestor (or, the node itself) of all the non-revoked nodes. Then it computes the t- component of the decryption key using the polynomials of all the nodes in that set.

- **Decryption Key Generation** $\mathcal{DK}(sk_\omega, ku_t)$:
  Parse sk_ω as $\{(i, D_i, d_i)\}_{i \in \mathsf{I}}$, ku_t as $\{(j, E_j, e_j)\}_{j \in \mathsf{J}}$ for some set of nodes I, J.

  $\forall (i, D_i, d_i) \in sk_\omega, (j, E_j, e_j) \in ku_t$
  If $\exists (i, j)$ s.t. $i = j$ then $dk_{\omega,t} \leftarrow (D_i, E_j, d_i, e_j)$
  Else (if sk_ω and ku_t don't have any node in common) then $dk_{\omega,t} \leftarrow \perp$.
  Return $dk_{\omega,t}$.

  // Above we can drop the subscripts i, j since they are equal, i.e. $dk_{\omega,t} = (D, E, d, e)$. The algorithm finds components of sk_ω and ku_t which were computed on the same polynomial.

- **Encryption** $\mathcal{E}(pk, \omega, t, m)$:
  Parse pk as $(g, g_1, g_2, h_1, h_2, h_3)$.
  $z \xleftarrow{\$} \mathbb{Z}_p$; $c_1 \leftarrow m \cdot \mathbf{e}(g_1, g_2)^z$; $c_2 \leftarrow g^z$;
  $c_\omega \leftarrow H_{g_2,\mathsf{J},h_1,h_2,h_3}(\omega)^z$; $c_t \leftarrow H_{g_2,\mathsf{J},h_1,h_2,h_3}(t)^z$.
  Return $c = (\omega, t, c_\omega, c_t, c_1, c_2)$.

  // The Encryption algorithm is essentially the same as that of Fuzzy IBE.

- **Decryption** $\mathcal{D}(dk_{\omega,t}, c)$:
  Parse $dk_{\omega,t}$ as (D, E, d, e), c as $(\omega, t, c_\omega, c_t, c_1, c_2)$.
  $m \leftarrow c_1 \left(\frac{\mathbf{e}(d,c_\omega)}{\mathbf{e}(D,c_2)}\right)^{\frac{t}{t-\omega}} \left(\frac{\mathbf{e}(e,c_t)}{\mathbf{e}(E,c_2)}\right)^{\frac{\omega}{\omega-t}}$.
  Return m.

  // The decryption algorithm is essentially the same as that of Fuzzy IBE.

- **Revocation** $\mathcal{R}(\omega, t, rl, st)$:
  For all nodes v associated with identity ω add (v, t) to rl.
  Return rl.

CONSISTENCY. If identity ω was not revoked before or, at time t, then we will show that $\mathcal{D}(dk_{\omega,t}, c) = m$ where $dk_{\omega,t}, m$ and c are computed as per the consistency requirement in Section 3.1.

From the definition of KUNodes we see that if ω was not revoked before or, at t then the set of nodes output by KUNodes has one ancestor (or, the node itself) of the leaf node associated with ω which implies that there will be a common node in sk_ω and ku_t and hence $\mathcal{DK}$ will not output $\perp$. Now from the above construction we have that for $a, a_x, z, r_\omega, r_t \in \mathbb{Z}_p$:

$g, g_2, h_1, h_2, h_3 \in \mathbb{G}$, $g_1 = g^a$
$dk_{\omega,t} = (D, E, d, e)$, where $D = g_2^{a_x\omega+a} H_{g_2,\mathsf{J},h_1,h_2,h_3}(\omega)^{r_\omega}$,
$E = g_2^{a_x t+a} H_{g_2,\mathsf{J},h_1,h_2,h_3}(t)^{r_t}$, $d = g^{r_\omega}$, $e = g^{r_t}$,
$c = (\omega, t, c_\omega, c_t, c_1, c_2)$, where $c_\omega = H_{g_2,\mathsf{J},h_1,h_2,h_3}(\omega)^z$,
$c_t = H_{g_2,\mathsf{J},h_1,h_2,h_3}(t)^z$, $c_1 = m \cdot \mathbf{e}(g_1, g_2)^z$, $c_2 = g^z$.

So, $\mathcal{D}(dk_{\omega,t}, c)$

$$
\begin{aligned}
&= c_1 \left(\frac{\mathbf{e}(d, c_\omega)}{\mathbf{e}(D, c_2)}\right)^{\frac{t}{t-\omega}} \left(\frac{\mathbf{e}(e, c_t)}{\mathbf{e}(E, c_2)}\right)^{\frac{\omega}{\omega-t}} \\
&= m \cdot \mathbf{e}(g_1, g_2)^z \\
&\times \left(\frac{\mathbf{e}(g^{r_\omega}, H_{g_2,\mathsf{J},h_1,h_2,h_3}(\omega)^z)}{\mathbf{e}(g_2^{a_x\omega+a} H_{g_2,\mathsf{J},h_1,h_2,h_3}(\omega)^{r_\omega}, g^z)}\right)^{\frac{t}{t-\omega}} \\
&\times \left(\frac{\mathbf{e}(g^{r_t}, H_{g_2,\mathsf{J},h_1,h_2,h_3}(t)^z)}{\mathbf{e}(g_2^{a_x t+a} H_{g_2,\mathsf{J},h_1,h_2,h_3}(t)^{r_t}, g^z)}\right)^{\frac{\omega}{\omega-t}} \\
&= m \cdot \mathbf{e}(g_1, g_2)^z \left(\frac{1}{\mathbf{e}(g_2^{a_x\omega+a}, g^z)}\right)^{\frac{t}{t-\omega}} \\
&\times \left(\frac{1}{\mathbf{e}(g_2^{a_x t+a}, g^z)}\right)^{\frac{\omega}{\omega-t}} \\
&= m \cdot \mathbf{e}(g_1, g_2)^z \\
&\times \left(\frac{1}{\mathbf{e}(g_2^{(a_x\omega+a)(\frac{t}{t-\omega})+(a_x t+a)(\frac{\omega}{\omega-t})}, g^z)}\right) \\
&= m \cdot \mathbf{e}(g_1, g_2)^z \frac{1}{\mathbf{e}(g_2^a, g^z)} = m \cdot \mathbf{e}(g_1, g_2)^z \frac{1}{\mathbf{e}(g_2, g_1)^z} \\
&= m. \ \blacksquare
\end{aligned}
$$

REMARKS. The function KUNodes needs to be executed only when rl has changed, so key authority can store the output of KUNodes and use it until rl changes. If the number of users exceeds n, the capacity of the current tree, it is possible to extend the tree and permit n more users as follows. Take an "empty" tree of the same size and connect the roots of the current and new trees to the new parent root node. Now the combined tree has $2n$ leaf nodes, and new users can be accommodated. Each user will need an additional private key component computed on the polynomial of the new root node. This new private key component can be encrypted (under the corresponding identity and time) and published.

EFFICIENCY. We first analyze communication and time complexity of key authority in computing and publishing key updates as a function of the number of users n and number of revoked users r.

	$r = 0$	$1 < r \leq n/2$	$n/2 < r \leq n$
BF [6]	$O(n)$	$O(n - r)$	$O(n - r)$
Revocable IBE	$O(1)$	$O(r \log(\frac{n}{r}))$	$O(n - r)$

Table 1: Key update complexity comparison. Above n is the total number of users and r is the number of revoked users.

We compare *the worst case* complexity of our scheme with that of the general revocation solution suggested by Boneh-Franklin [6] that we outlined in the Introduction. Table 1 summarizes the results. The complexity analysis for our construction follows directly from Theorem 1 of [1], as the number of necessary key updates in our scheme corresponds to the number of nodes returned by function KUNodes, and a similar function on the binary tree was used in [1].

As the table shows, our scheme represents a significant improvement over the Boneh-Franklin solution for small values of r. For larger values of r (especially as it reaches close to n), this advantage is lost. We however note that as r becomes large, our scheme can be "reset" to keep key update efficient (by running the setup algorithm again which will make the revocation list empty and releasing new private keys for only non-revoked users).

In terms of encryption and decryption, our construction is slightly less efficient than the existing IBE schemes. E.g. the decryption algorithms of IBEs by Waters [25] and Boneh-Boyen [3] require 2 pairing computations (the slowest computation compared to group operations and exponentiations), and our scheme requires 4. Encryption in the schemes of [25, 3] is dominated by 3 and 4 exponentiations, while our scheme uses 12. We chose Waters and Boneh-Boyen constructions for comparison because they are the most efficient IBE schemes secure in standard (RO devoid) model under standard assumptions. This may be a reasonable price to pay for the significant improvement in key-update efficiency, which may become a bottleneck for a large number of users. We note that the size of secret keys is larger in our scheme, a user needs to store up to $3h = 3 \log n$ group elements.

We note that using the suggestion from [22], efficiency of our scheme, and in particular, its encryption algorithm, can be improved, if a hash function is used in place of the function H. Security analysis in this case will need to rely on the random oracle (RO) model [2]. This will improve the number of exponentiations in encryption to 4 while the decryption algorithm will still be dominated by 4 paring operations. In contrast, the cost of encryption and decryption in the Boneh-Franklin scheme [6] is dominated by one pairing each.

Security. Even though different users have their private keys computed on the same polynomial this does not introduce insecurity in $\mathcal{RIBE}$ as opposed to Fuzzy IBE. In our scheme collusion among different users is possible, however such collusion is not useful. No matter how many revoked users try to collude, they will still be unable to decrypt a ciphertext for a new time period, as they cannot obtain the necessary decryption key component. One might be tempted to reduce the security of $\mathcal{RIBE}$ to the security of Fuzzy IBE since, after all $\mathcal{RIBE}$ uses Fuzzy IBE as its base construction. However, we want to point out that such a straightforward reduction of security in a black box manner does not seem possible. The main reason for this is that in Fuzzy IBE each time key generation algorithm is run it chooses a random polynomial and then computes the key using that polynomial however, in $\mathcal{RIBE}$ it is essential that private key and key update be computed on some fixed polynomials.

Security of $\mathcal{RIBE}$ is based on the hardness of *decisional bilinear Diffie-Hellman* (DBDH) prob-

lem, which we now recall.

Definition 4.2 [Decisional bilinear Diffie-Hellman] Let $\mathcal{G}$ be a prime order bilinear group generator. The *decisional bilinear Diffie-Hellman (DBDH)* problem is said to be hard for $\mathcal{G}$ if for every efficient adversary A its advantage $\mathbf{Adv}^{\text{dbdh}}_{\mathcal{G},A}(k)$ defined as

$$\Pr\left[\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},A}(1^\kappa)=1\right]-\Pr\left[\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},A}(1^\kappa)=1\right]$$

is a negligible function in κ, and where the experiments are as follows:

Experiment $\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},A}(1^\kappa)$
$(\tilde{\mathbb{G}},p,g)\xleftarrow{\$}\mathcal{G}(1^\kappa)\ ;\ x,y,z\xleftarrow{\$}\mathbb{Z}_p$
$X\leftarrow g^x\ ;\ Y\leftarrow g^y\ ;\ Z\leftarrow g^z\ ;\ W\leftarrow \mathbf{e}(g,g)^{xyz}$
$d\xleftarrow{\$}A(1^\kappa,\tilde{\mathbb{G}},p,g,X,Y,Z,W)$
Return d

Experiment $\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},A}(1^\kappa)$
$(\tilde{\mathbb{G}},p,g)\xleftarrow{\$}\mathcal{G}(1^\kappa)\ ;\ x,y,z,w\xleftarrow{\$}\mathbb{Z}_p$
$X\leftarrow g^x\ ;\ Y\leftarrow g^y\ ;\ Z\leftarrow g^z\ ;\ W\leftarrow \mathbf{e}(g,g)^{w}$
$d\xleftarrow{\$}A(1^\kappa,\tilde{\mathbb{G}},p,g,X,Y,Z,W)$
Return d ▌

We now state the security result.

Theorem 4.3 Let $\mathcal{G}$ be a prime order bilinear group generator and $\mathcal{RIBE}[\mathcal{G}]=(\mathcal{S},\mathcal{SK},\mathcal{KU},\mathcal{DK},\mathcal{E},\ \mathcal{D},\mathcal{R})$ be the associated Revocable IBE scheme defined by Construction 4.1. Then for any adversary $\mathcal{A}$ attacking sRID-CPA security of $\mathcal{RIBE}$ with n users, whose running time is $t_{\mathcal{A}}$ and who asks q_p private key generation queries, q_k key update generation queries and q_r revocation queries, there exists an adversary $\mathcal{B}$ solving DBDH problem for $\mathcal{G}$ such that

$$\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{A},n}(\kappa)\leq 4\cdot\mathbf{Adv}^{\text{dbdh}}_{\mathcal{G},\mathcal{B}}(\kappa)\ , \tag{1}$$

where the running time of $\mathcal{B}$, $t_{\mathcal{B}}=t_{\mathcal{A}}+O(\kappa^3)$.

Proof: We construct an adversary $\mathcal{B}$ for the DBDH problem associated with $\mathcal{G}$. $\mathcal{B}$ gets $(1^\kappa,\tilde{\mathbb{G}},p,g,$ $X,\ Y,Z,W)$ as input and it has to return a bit d. It is going to use $\mathcal{A}$.

$\mathcal{B}(1^\kappa, \tilde{\mathbb{G}}, p, g, X, Y, Z, W)$:

$(\omega^*, t^*, state) \xleftarrow{\$} \mathcal{A}(1^\kappa)$

$g_1 \leftarrow X,\ g_2 \leftarrow Y$

Pick random second-degree polynomials $f(x), u(x)$ with coefficients in $\mathbb{Z}_p$, s.t. $u(x) = -x^2$ for $x = \omega^*, t^*$, o.w. $u(x) \neq -x^2$

For $i = 1, 2, 3 : h_i \leftarrow g_2^{u(i)} g^{f(i)}$

$pk \leftarrow (g, g_1, g_2, h_1, h_2, h_3)$

Let rl be an empty set and T be a binary tree with at least n leaf nodes.

Pick a leaf node v^* from T and a random bit rev

Run $\mathcal{A}$ and answer its queries to the $\mathcal{SK}(\cdot), \mathcal{KU}(\cdot)$, $\mathcal{R}(\cdot,\cdot)$ oracles using the subroutines defined.

$(m_0, m_1, state') \xleftarrow{\$} \mathcal{A}^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot)}(1^\kappa, pk, state)$

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

$c_1^* \leftarrow m_b \cdot W,\ c_2^* \leftarrow Z,\ c_\omega^* \leftarrow Z^{f(\omega^*)},\ c_t^* \leftarrow Z^{f(t^*)}$

$c^* \leftarrow (\omega^*, t^*, c_\omega^*, c_t^*, c_1^*, c_2^*)$

$d \xleftarrow{\$} \mathcal{A}^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot)}(1^\kappa, pk, c^*, state')$

If any of the oracles abort return 1

Else if $b = d$ return 1 else return 0

Now, we present the subroutines for answering the oracle queries. These subroutines use the functions that we define here. For $i, j, l, r' \in \mathbb{Z}_p,\ S = \{0, j\}$ define

$$F_1(i,j,l,r') \stackrel{\text{def}}{=} g_2^{l\Delta_{j,S}(i)} \left(g_1^{\frac{-f(i)}{i^2+u(i)}} \left(g_2^{i^2+u(i)} g^{f(i)} \right)^{r'} \right)^{\Delta_{0,S}(i)},$$

$$F_2(i,r') \stackrel{\text{def}}{=} \left(g_1^{\frac{-1}{i^2+u(i)}} g^{r'} \right)^{\Delta_{0,S}(i)}.$$

$\mathcal{SK}(\omega)$:

If $rev = 0$ and $\omega = \omega^*$ then abort

If $rev = 1$ and $\omega = \omega^*$ then:

$v \leftarrow v^*$,

$\forall x \in \mathsf{Path}(v)\ r_x \xleftarrow{\$} \mathbb{Z}_p$

if $\nexists l_x$ then $l_x \xleftarrow{\$} \mathbb{Z}_p$ and store l_x in node x

$D_x \leftarrow g_2^{l_x} H_{g_2,h_1,h_2,h_3}(\omega^*)^{r_x},\ d_x \leftarrow g^{r_x}$

If $rev = 0$ and $\omega \neq \omega^*$ then:

Pick an unassigned leaf node v from T and store ω in node v

$\forall x \in \mathsf{Path}(v)\ r'_x \xleftarrow{\$} \mathbb{Z}_p$,

if $\nexists l_x$ then $l_x \xleftarrow{\$} \mathbb{Z}_p$ and store l_x in node x

$i \leftarrow \omega, j \leftarrow t^*, l \leftarrow l_x, r' \leftarrow r'_x$

$D_x \leftarrow F_1(i,j,l,r'),\ d_x \leftarrow F_2(i,r')$

If $rev = 1$ and $\omega \neq \omega^*$ then:

Pick an unassigned leaf node v from T and store ω in node v

$\forall x \in \mathsf{Path}(v)\ r'_x \xleftarrow{\$} \mathbb{Z}_p$,

if $\nexists l_x$ then $l_x \xleftarrow{\$} \mathbb{Z}_p$ and store l_x in node x
$i \leftarrow \omega, l \leftarrow l_x, r' \leftarrow r'_x$
$\forall x \in (\mathsf{Path}(v) \setminus \mathsf{Path}(v^*))$
$j \leftarrow t^*$
$D_x \leftarrow F_1(i, j, l, r'),\ d_x \leftarrow F_2(i, r')$
$\forall x \in (\mathsf{Path}(v) \cap \mathsf{Path}(v^*))$
$j \leftarrow \omega^*$
$D_x \leftarrow F_1(i, j, l, r'),\ d_x \leftarrow F_2(i, r')$
Return $sk_\omega = \{(x, D_x, d_x)\}_{x \in \mathsf{Path}(v)}$

$\mathcal{R}(\omega, t)$:
For all leaf nodes v associated with identity ω, add (v, t) to rl

$\mathcal{KU}(t)$:
If $rev = 1$ and $t = t^*$ and $\forall t \leq t^*$ we have that $(\omega^*, t) \notin rl$, then abort
Else if $t = t^*$ then:
$\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t), r_x \xleftarrow{\$} \mathbb{Z}_p$
$E_x \leftarrow g_2^{l_x} H_{g_2,h_1,h_2,h_3}(t^*)^{r_x},\ e_x \leftarrow g^{r_x}$
If $rev = 1$ and $t \neq t^*$ then:
$\forall x \in (\mathsf{KUNodes}(\mathsf{T}, rl, t) \setminus \mathsf{Path}(v^*))$
$r'_x \xleftarrow{\$} \mathbb{Z}_p, i \leftarrow t, j \leftarrow t^*, l \leftarrow l_x, r' \leftarrow r'_x$
$E_x \leftarrow F_1(i, j, l, r'),\ e_x \leftarrow F_2(i, r')$
$\forall x \in (\mathsf{KUNodes}(\mathsf{T}, rl, t) \cap \mathsf{Path}(v^*))$
$r'_x \xleftarrow{\$} \mathbb{Z}_p, i \leftarrow t, j \leftarrow \omega^*, l \leftarrow l_x, r' \leftarrow r'_x$
$E_x \leftarrow F_1(i, j, l, r'),\ e_x \leftarrow F_2(i, r')$
If $rev = 0$ and $t \neq t^*$ then:
$\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)$
$r'_x \xleftarrow{\$} \mathbb{Z}_p, i \leftarrow t, j \leftarrow t^*, l \leftarrow l_x, r' \leftarrow r'_x$
$E_x \leftarrow F_1(i, j, l, r'),\ e_x \leftarrow F_2(i, r')$
Return $ku_t = \{(x, E_x, e_x)\}_{x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)}$

Analysis. $f(x)$ being a random polynomial ensures that h_1, h_2, h_3 are random so, pk has the right distribution. For $i = 1, 2, 3$ $h_i = g_2^{u(i)} g^{f(i)}$ which implies $H_{g_2,h_1,h_2,h_3}(x) = g_2^{x^2+u(x)} g^{f(x)}$. Since $x = \omega^*, t^*$ $u(x) = -x^2$, then $H_{g_2,h_1,h_2,h_3}(\omega^*) = g^{f(\omega^*)}, H_{g_2,h_1,h_2,h_3}(t^*) = g^{f(t^*)}$. If $W = \mathbf{e}(g, g)^{xyz}$, then $c_1^* = m_b \cdot W = m_b \cdot \mathbf{e}(g_1, g_2)^z, c_{\omega^*} = Z^{f(\omega^*)} = g^{zf(\omega^*)} = H_{g_2,h_1,h_2,h_3}(\omega^*)^z, c_{t^*} = Z^{f(t^*)} = g^{zf(t^*)} = H_{g_2,h_1,h_2,h_3}(t^*)^z$. So, if $\mathcal{B}$ is in $\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(\kappa)$ then c^* is a well-formed ciphertext of m_b. Otherwise, if $W = \mathbf{e}(g, g)^w$ for a random w, then $c_1^* = m_b \cdot W = m_b \cdot \mathbf{e}(g, g)^w$, so w being random and independent from x, y, z ensures that c_1^* is also random and thus c^* hides bit b information-theoretically. In what follows let $g_1 = g^a$ (to simplify notation).

Case 1. $rev = 1, \omega = \omega^*$ in $\mathcal{SK}(\omega)$ oracle simulation:
Define $a_x = \frac{1}{\omega^*}(l_x - a)$. Then $D_x = g_2^{a_x\omega^*+a} H_{g_2,h_1,h_2,h_3}(\omega^*)^{r_x}, d_x = g^{r_x}$. So, sk_ω has the right

distribution.

Case 2. $rev = 0, \omega \neq \omega^*$ in $\mathcal{SK}(\omega)$ oracle simulation:
Define $a_x = \frac{1}{t^*}(l_x - a),\ r_x = (r'_x - \frac{a}{\omega^2+u(\omega)})\Delta_{0,S}(\omega)$. Then

$$\begin{aligned} F_2(i,r') &= \left(g_1^{\frac{-1}{i^2+u(i)}} g^{r'}\right)^{\Delta_{0,S}(i)} \\ &= \left(g^{r'-\frac{a}{i^2+u(i)}}\right)^{\Delta_{0,S}(i)} \\ &= \left(g^{r'_x-\frac{a}{\omega^2+u(\omega)}}\right)^{\Delta_{0,S}(\omega)} \\ &= g^{r'_x} \end{aligned}$$

and

$$\begin{aligned} F_1(i,j,l,r') &= g_2^{l\Delta_{j,S}(i)} \left(\left(g_1^{\frac{-f(i)}{i^2+u(i)}}\right)\left(g_2^{i^2+u(i)} g^{f(i)}\right)^{r'}\right)^{\Delta_{0,S}(i)} \\ &= g_2^{l\Delta_{j,S}(i)} \left(\left(g^{\frac{-af(i)}{i^2+u(i)}}\right)\left(g_2^{i^2+u(i)} g^{f(i)}\right)^{r'}\right)^{\Delta_{0,S}(i)} \\ &= g_2^{l\Delta_{j,S}(i)} \left(g_2^a \left(g_2^{i^2+u(i)} g^{f(i)}\right)^{\frac{-a}{i^2+u(i)}} \left(g_2^{i^2+u(i)} g^{f(i)}\right)^{r'}\right)^{\Delta_{0,S}(i)} \\ &= g_2^{l\Delta_{j,S}(i)} \left(g_2^a \left(g_2^{i^2+u(i)} g^{f(i)}\right)^{r'-\frac{a}{i^2+u(i)}}\right)^{\Delta_{0,S}(i)} \\ &= g_2^{l\Delta_{j,S}(i)+a\Delta_{0,S}(i)} \left(g_2^{i^2+u(i)} g^{f(i)}\right)^{\left(r'-\frac{a}{i^2+u(i)}\right)\Delta_{0,S}(i)} \\ &= g_2^{l\Delta_{j,S}(i)+a\Delta_{0,S}(i)} H_{g_2,h_1,h_2,h_3}(i)^{\left(r'-\frac{a}{i^2+u(i)}\right)\Delta_{0,S}(i)} \\ &= g_2^{l_x\Delta_{j,S}(\omega)+a\Delta_{0,S}(\omega)} H_{g_2,h_1,h_2,h_3}(\omega)^{\left(r'_x-\frac{a}{\omega^2+u(\omega)}\right)\Delta_{0,S}(\omega)} \\ &= g_2^{a_x\omega+a} H_{g_2,h_1,h_2,h_3}(\omega)^{r_x}. \end{aligned}$$

So, sk_ω has the right distribution.

Case 3. $rev = 1$ and $\omega \neq \omega^*$. Similar arguments as above apply if we define
$a_x = \frac{1}{\omega^*}(l_x - a),\ r_x = (r'_x - \frac{a}{\omega^2+u(\omega)})\Delta_{0,S}(\omega)$ for nodes on the path of v^* and
$a_x = \frac{1}{t^*}(l_x - a),\ r_x = (r'_x - \frac{a}{\omega^2+u(\omega)})\Delta_{0,S}(\omega)$ for rest of the nodes.

Similar arguments as above apply for all cases in $\mathcal{KU}(t)$ oracle simulation if we define:

In case $rev = 0$
$a_x = \frac{1}{t^*}(l_x - a),\ r_x = (r'_x - \frac{a}{t^2+u(t)})\Delta_{0,S}(t)$ for all nodes.

And, in case $rev = 1$
$a_x = \frac{1}{\omega^*}(l_x - a),\ r_x = (r'_x - \frac{a}{t^2+u(t)})\Delta_{0,S}(t)$ for nodes on the path of v^* and
$a_x = \frac{1}{t^*}(l_x - a),\ r_x = (r'_x - \frac{a}{t^2+u(t)})\Delta_{0,S}(t)$ for rest of the nodes.

Note that we are defining a_x consistently in both oracle simulations, i.e. in case $rev = 0$, $a_x = \frac{1}{t^*}(l_x - a)$ for all nodes, and in case $rev = 1$, $a_x = \frac{1}{\omega^*}(l_x - a)$ for all nodes on the path from v^* to the root node, $a_x = \frac{1}{t^*}(l_x - a)$ for rest of the nodes. Also note that values l_x which are stored in node x, and identities ω which are stored in leaf nodes of tree T have the right distribution and also that they are modified only by $\mathcal{SK}(\omega)$ oracle. $\mathcal{KU}(t)$ and $\mathcal{R}(\omega, t)$ oracles do not modify them. Similarly, revocation list rl is modified only by $\mathcal{R}(\omega, t)$ oracle. $\mathcal{SK}(\omega)$ and $\mathcal{KU}(t)$ oracles do not modify them. Thus, oracles maintain the state and revocation list consistently and with right distribution.

Claim 4.4 Let $\mathsf{sreal}, \mathsf{srand}$ denote the events that none of the oracles abort in $\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)$, $\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)$ respectively. Then,

$$\Pr[\,\mathsf{sreal}\,] = \Pr[\,\mathsf{srand}\,] \geq \frac{1}{2}.$$

Proof: We will prove the claim in two parts. First we will show that $\Pr[\,\mathsf{sreal}\,] = \Pr[\,\mathsf{srand}\,]$ and then we will show that $\Pr[\,\mathsf{sreal}\,] \geq \frac{1}{2}$.

First part is easy to see. The probability that $\mathcal{SK}(\omega)$ and $\mathcal{KU}(t)$ oracles abort depends on the bit rev which is chosen independently from whether $\mathcal{B}$ is in $\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)$ or $\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)$. So, $\Pr[\,\mathsf{sreal}\,] = \Pr[\,\mathsf{srand}\,]$. Now it remains to show that $\Pr[\,\mathsf{sreal}\,] \geq \frac{1}{2}$.

Condition 3 of Definition 3.2 says that $\mathcal{SK}(\omega)$ oracle can be queried on ω^* only if $\mathcal{R}(\omega, t)$ oracle was queried on (ω^*, t) for any $t \leq t^*$. Thus, we have

$$\begin{aligned}
& \Pr[\,\omega = \omega^*\,] \leq \Pr[\,(\omega^*, t) \in rl,\ \forall\ t \leq t^*\,] \\
\Rightarrow\ & 1 - \Pr[\,\omega = \omega^*\,] \geq \Pr[\,(\omega^*, t) \notin rl,\ \forall\ t \leq t^*\,] \\
\Rightarrow\ & 1 - \Pr[\,\omega = \omega^*\,] \geq \Pr[\,(t = t^*) \wedge ((\omega^*, t) \notin rl,\ \forall\ t \leq t^*)\,].
\end{aligned}$$

We see that $\mathcal{SK}(\omega)$ oracle aborts if $rev = 0$ and $\omega = \omega^*$ and $\mathcal{KU}(t)$ oracle aborts if $rev = 1, t = t^*$ and $\forall t \leq t^*\ (\omega^*, t) \notin rl$. Thus,

$$\begin{aligned}
\Pr[\,\overline{\mathsf{sreal}}\,] &= \Pr[\,(rev = 0) \wedge (\omega = \omega^*)\,] \\
&\quad + \Pr[\,(rev = 1) \wedge (t = t^*) \wedge ((\omega^*, t) \notin rl,\ \forall\ t \leq t^*)\,] \\
&= \Pr[\,rev = 0\,] \cdot \Pr[\,\omega = \omega^*\,] \\
&\quad + \Pr[\,rev = 1\,] \cdot \Pr[\,(t = t^*) \wedge ((\omega^*, t) \notin rl,\ \forall\ t \leq t^*)\,] \\
&\leq \frac{1}{2}\Pr[\,\omega = \omega^*\,] + \frac{1}{2}(1 - \Pr[\,\omega = \omega^*\,]) \\
&\leq \frac{1}{2}.
\end{aligned}$$

Therefore, $\Pr[\,\mathsf{sreal}\,] \geq \frac{1}{2}$. ∎

∎

We have shown above that when $\mathcal{B}$ is in $\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)$ and none of the oracles abort, then $\mathcal{B}$ is simulating the exact experiment $\mathbf{Exp}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{A},n}(1^\kappa)$ for $\mathcal{A}$. So,

$$\Pr\left[\,\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(\kappa)=1 \mid \mathsf{sreal}\,\right] \geq \Pr\left[\,\mathbf{Exp}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{A},n}(1^\kappa)=1\,\right].$$

When $\mathcal{B}$ is in $\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)$ and none of the oracles abort then as explained earlier bit b is information-theoretically hidden from $\mathcal{A}$. So,

$$\Pr\left[\,\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(\kappa)=1 \mid \mathsf{srand}\,\right] \leq \frac{1}{2}.$$

Also, since $\mathcal{B}$ outputs 1 when either of the oracles aborts, so

$$\begin{aligned}\Pr\left[\,\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1 \mid \overline{\mathsf{sreal}}\,\right]&=1,\\ \Pr\left[\,\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1 \mid \overline{\mathsf{srand}}\,\right]&=1\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{Adv}^{\text{dbdh}}_{\mathcal{G},\mathcal{B}}(\kappa) &= \Pr\left[\,\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1\,\right]-\Pr\left[\,\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1\,\right]\\ &= \Pr[\,\mathsf{sreal}\,]\cdot\Pr\left[\,\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1|\mathsf{sreal}\,\right]\\ &\quad+\Pr[\,\overline{\mathsf{sreal}}\,]\cdot\Pr\left[\,\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1|\overline{\mathsf{sreal}}\,\right]\\ &\quad-\Pr[\,\mathsf{srand}\,]\cdot\Pr\left[\,\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1|\mathsf{srand}\,\right]\\ &\quad-\Pr[\,\overline{\mathsf{srand}}\,]\cdot\Pr\left[\,\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1|\overline{\mathsf{srand}}\,\right]\\ &\geq \frac{1}{2}\cdot\Big(\Pr\left[\,\mathbf{Exp}^{\text{dbdh-real}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1|\mathsf{sreal}\,\right]\\ &\quad-\Pr\left[\,\mathbf{Exp}^{\text{dbdh-rand}}_{\mathcal{G},\mathcal{B}}(1^\kappa)=1|\mathsf{srand}\,\right]\Big)\\ &\geq \frac{1}{2}\cdot\left(\Pr\left[\,\mathbf{Exp}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{A},n}(1^\kappa)=1\,\right]-\frac{1}{2}\right)\\ &\geq \frac{1}{4}\cdot\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{A},n}(\kappa)\,.\end{aligned}$$

Finally we observe that $t_{\mathcal{B}}=t_{\mathcal{A}}+O(\kappa^3)$ as $\mathcal{B}$ does only a constant number of modulo exponentiations and multiplications and picks a constant number of random group elements (recall that by convention $t_{\mathcal{A}}$ includes the time of the experiment including computations done by the oracles.˙ ∎

5 Addressing CCA Security

We suggest two ways to construct RIBE schemes that resist chosen-ciphertext attacks. Our first solution is a modification of our main construction. Our second solution is generic in that it is based on any sRID-CPA secure scheme, though CCA security relies on the RO model.

RIBE$_{CCA}$ Construction. We combine the ideas of [4] (used there for a different problem of constructing an IND-CCA public-key encryption scheme) with the error-tolerance property of Fuzzy

IBE to modify our Revocable IBE scheme. Changes are mainly in the encryption and decryption algorithms. We employ a strongly-unforgeable one-time signature scheme (cf. [4] that recalls the primitive and its security definition). The setup algorithm of the new scheme is very similar to the one in Fuzzy IBE where 2 out of 3 attributes of ciphertexts should match with those of the decryption key. The private key generation and key update generation algorithms are very similar to those of $\mathcal{RIBE}$ except that we now use second-degree polynomials as opposed to first-degree polynomials in $\mathcal{RIBE}$. The encryption algorithm runs the key generation algorithm of one-time signature to obtain a signing key and verification key and then encrypts the message with three attributes: identity, time and verification key. Then it signs the resulting intermediate ciphertext using the signing key. The decryption algorithm verifies the signature, and that ciphertext is properly formed (by using a ciphertext sanity check due to [13]) before decrypting.

Let $\mathcal{G}$ be a bilinear group generator and $\mathcal{OTS} = (\mathsf{SGen}, \mathsf{Sign}, \mathsf{Ver})$ be a strongly-unforgeable one-time signature scheme. Let $\mathcal{RIBE}[\mathcal{G}] = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ be the scheme of Construction 4.1. We define $\mathcal{RIBE}_{CCA}[\mathcal{G}, \mathcal{OTS}] = (\mathcal{S}', \mathcal{SK}', \mathcal{KU}', \mathcal{DK}, \mathcal{E}', \mathcal{D}', \mathcal{R})$ by specifying the differences from $\mathcal{RIBE}$. Here we require that identities, time periods and the verification keys for the one-time signature output by SGen are mapped to distinguished elements in $\mathbb{Z}_p^*$ (e.g., by pre-pending "00", "01" and "11" to strings of these types and then using a collision-resistant hash function that maps $\{0,1\}^*$ to $\mathbb{Z}_p^*$. Let J be $\{1,2,3,4\}$.

- **Setup** $\mathcal{S}'(\kappa, n)$:
  Everything is the same as in $\mathcal{S}$ except that pk has an additional element $h_4 \xleftarrow{\$} \mathbb{G}$ and $pk = (g, g_1, g_2, h_1, h_2, h_3, h_4)$.

- **Private Key Generation** $\mathcal{SK}'(pk, mk, \omega, st)$:
  Everything is the same as in $\mathcal{SK}$ except that now we pick a random second-degree polynomial $q_x(y)$ with coefficients in $\mathbb{Z}_p$ and the same restriction that $q_x(0) = a$. Parse pk as $(g, g_1, g_2, h_1, h_2, h_3, h_4)$, mk as a, st as T. Pick an unassigned leaf node v from T and store ω in that node.

  $\forall x \in \mathsf{Path}(v)$
  if q_x is undefined, then pick a random second-degree polynomial q_x,
  s.t. $q_x(0) = a$ and store q_x in node x
  $r_x \xleftarrow{\$} \mathbb{Z}_p$; $D_x \leftarrow g_2^{q_x(\omega)} H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(\omega)^{r_x}$; $d_x \leftarrow g^{r_x}$
  Return $sk_\omega = \{(x, D_x, d_x)\}_{x \in \mathsf{Path}(v)},\ st$.

- **Key Update Generation** $\mathcal{KU}'(pk, mk, t, rl, st)$:
  Parse pk as $(g, g_1, g_2, h_1, h_2, h_3, h_4)$, mk as a, st as T.

  $\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)$
  $r_x \xleftarrow{\$} \mathbb{Z}_p$; $E_x \leftarrow g_2^{q_x(t)} H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(t)^{r_x}$; $e_x \leftarrow g^{r_x}$
  Return $ku_t = \{(x, E_x, e_x)\}_{x \in \mathsf{KUNodes}(\mathsf{T},rl,t)}$.

- **Encryption** $\mathcal{E}'(pk, \omega, t, m)$:
  Parse pk as $(g, g_1, g_2, h_1, h_2, h_3, h_4)$
  $(sigk, vk) \xleftarrow{\$} \mathsf{SGen}(1^\kappa)$.
  $z \xleftarrow{\$} \mathbb{Z}_p$; $c_1 \leftarrow m \cdot \mathbf{e}(g_1, g_2)^z$; $c_2 \leftarrow g^z$; $c_\omega \leftarrow H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(\omega)^z$

$c_t \leftarrow H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(t)^z$; $c_{vk} \leftarrow H_{g_2,h_1,h_2,h_3,h_4}(vk)^z$
$c \leftarrow (\omega, t, c_\omega, c_t, c_{vk}, c_1, c_2)$
$\sigma \leftarrow \mathsf{Sign}(sigk, c)$
Return $\tilde{c} = (c, \sigma, vk)$.

- **Decryption** $\mathcal{D}'(dk_{\omega,t}, \tilde{c})$:
  Parse $dk_{\omega,t}$ as (D, E, d, e), $\tilde{c}$ as $(c = (\omega, t, c_\omega, c_t, c_{vk},\ c_1, c_2), \sigma, vk)$
  If $\mathsf{Ver}(vk, c, \sigma) \neq 1$ then return $\perp$.
  Else pick $r_1, r_2, r_3 \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
  If $\mathbf{e}(c_2, H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(\omega)^{r_1} \cdot H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(t)^{r_2} \times H_{g_2,\mathsf{J},h_1,h_2,h_3,h_4}(vk)^{r_3}) \neq$
  $\mathbf{e}(g, c_\omega^{r_1} c_t^{r_2} c_{vk}^{r_3})$ then return $\perp$
  Else $m \leftarrow c_1 \left(\frac{\mathbf{e}(d,c_\omega)}{\mathbf{e}(D,c_2)}\right)^{\frac{t}{t-\omega}} \left(\frac{\mathbf{e}(e,c_t)}{\mathbf{e}(E,c_2)}\right)^{\frac{\omega}{\omega-t}}$
  Return m.

One can verify that consistency follows directly from consistency of $\mathcal{OTS}$ and $\mathcal{RIBE}$. ∎

RIBE$_{CCA}$ Security. We claim the following.

Theorem 5.1 Let $\mathcal{G}$ be a prime order bilinear group generator, $\mathcal{OTS} = (\mathsf{SGen}, \mathsf{Sign}, \mathsf{Ver})$ be a one-time signature scheme and $\mathcal{RIBE}_{CCA}[\mathcal{G}, \mathcal{OTS}] = (\mathcal{S}',\ \mathcal{SK}', \mathcal{KU}', \mathcal{DK}, \mathcal{E}',\ \mathcal{D}',\ \mathcal{R})$ be the associated Revocable IBE scheme as per construction above. Then for any adversary $\mathcal{A}$ attacking sRID-CCA security of $\mathcal{RIBE}$ with n users, whose running time is $t_\mathcal{A}$ and who asks q_p private key generation queries, q_k key update generation queries, q_r revocation queries and q_d decryption queries, there exist an adversary $\mathcal{B}$ solving DBDH problem for $\mathcal{G}$ and an adversary F attacking strong unforgeability of $\mathcal{OTS}$ such that

$$\mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{RIBE},\mathcal{A},n}(\kappa) \leq 4 \cdot \mathbf{Adv}^{\text{dbdh}}_{\mathcal{G},\mathcal{B}}(\kappa) + \mathbf{Adv}^{\text{suf}-\text{ots}}_{\mathcal{OTS},F}(\kappa)\ , \tag{2}$$

where the last term refers to the advantage of F breaking strong unforgeability of $\mathcal{OTS}$ (cf. [4] for the definition) and $t_\mathcal{B} \approx t_F \approx t_\mathcal{A} + q_d(\frac{2t_p}{\log n}) + O(k^3)$.

We provide some intuition before giving the actual proof. It is not hard to show that $\mathcal{RIBE}_{CCA}$ is sRID-CPA secure, the security proof is very similar to the proof of Theorem 4.3. Even though a ciphertext is encrypted under an additional attribute: the verification key, the key authority never issues the corresponding decryption key component. To show that $\mathcal{RIBE}_{CCA}$ is also sRID-CCA secure, the simulator (the DBDH adversary) needs to simulate the decryption oracle. Using the arguments very similar to those used in [13] we can show that the randomized check in the decryption algorithm that the simulator can perform as well, does guarantee with overwhelming probability that a ciphertext was formed correctly (according to the encryption algorithm). If the adversary queries a ciphertext whose verification key component is the same as that of the challenge ciphertext, then the decryption query cannot be answered correctly, but in this case one can construct an adversary breaking security of the one-time signature scheme. If the verification keys are different and the ciphertext passes the randomized check, then the simulator can generate the decryption key corresponding to identity and verification key of the queried ciphertext, and such a decryption key will successfully decrypt the ciphertext because we know from the randomized check that the queried ciphertext is a well-formed ciphertext. Generating such a decryption key is possible because the verification key is different from the challenge verification key, and following the proof of security of Fuzzy IBE, it is possible for the simulator to generate valid keys for a set of

attributes if they overlap with the challenge set of attributes in fewer than the threshold number of attributes.

Proof: We construct an adversary $\mathcal{B}$ for the DBDH problem associated with $\mathcal{G}$. $\mathcal{B}$ gets $(1^\kappa, \tilde{\mathbb{G}}, p, g, X, Y, Z, W)$ as input and it has to return a bit d. It is going to use $\mathcal{A}$.

$\mathcal{B}(1^\kappa, \tilde{\mathbb{G}}, p, g, X, Y, Z, W)$:
$(\omega^*, t^*, state) \xleftarrow{\$} \mathcal{A}(1^\kappa)$
$g_1 \leftarrow X,\ g_2 \leftarrow Y$
$sigk^*, vk^* \xleftarrow{\$} \mathsf{SGen}(1^\kappa),\ uk^* \leftarrow 01 \| vk^*,\ l_{uk^*} \xleftarrow{\$} \mathbb{Z}_p$
Pick random third-degree polynomials $f(x), u(x)$ with coefficients in $\mathbb{Z}_p$,
s.t. $u(x) = -x^3$ for $x = \omega^*, t^*, uk^*$, o.w. $u(x) \neq -x^3$
For $i = 1, 2, 3, 4 : h_i \leftarrow g_2^{u(i)} g^{f(i)}$
$pk = (g, g_1, g_2, h_1, h_2, h_3, h_4)$
Let rl be an empty set and T be a binary tree with at least n leaf nodes
Pick a leaf node v^* from T and a random bit rev.
Run $\mathcal{A}$ and answer its queries to the $\mathcal{SK}'(\cdot), \mathcal{KU}'(\cdot), \mathcal{R}(\cdot,\cdot), \mathcal{D}'(\cdot)$ oracles using subroutines defined below
$(m_0, m_1, state') \xleftarrow{\$} \mathcal{A}^{\mathcal{SK}'(\cdot), \mathcal{KU}'(\cdot), \mathcal{R}(\cdot,\cdot), \mathcal{D}'(\cdot)}(1^\kappa, pk, state)$
$b \xleftarrow{\$} \{0, 1\}$
$c^*_\omega \leftarrow Z^{f(\omega^*)},\ c^*_t \leftarrow Z^{f(t^*)}, c^*_{uk} \leftarrow Z^{f(uk^*)}, c^*_1 \leftarrow m_b \cdot W, c^*_2 \leftarrow Z$
$c^* \leftarrow (\omega^*, t^*, c^*_\omega, c^*_t, c^*_{uk}, c^*_1, c^*_2)$
$\sigma^* \leftarrow \mathsf{Sign}(sigk^*, c^*)$.
$\tilde{c}^* \leftarrow (c^*, \sigma^*, vk^*)$.
$d \xleftarrow{\$} \mathcal{A}^{\mathcal{SK}(\cdot), \mathcal{KU}(\cdot), \mathcal{R}(\cdot,\cdot), \mathcal{D}(\cdot)}(1^\kappa, pk, \tilde{c}^*, state')$
If any of the oracles abort return 1
Else if $b = d$ return 1 else return 0

Now, we present the subroutines for answering the oracle queries. Subroutines for $\mathcal{SK}'(\cdot), \mathcal{KU}'(\cdot), \mathcal{R}(\cdot,\cdot)$ oracles are very similar to those in the Proof of Theorem 4.3, but we give them here too for the sake of completeness. The oracles use the functions that we define here.

For $i, j, l, r' \in \mathbb{Z}_p,\ uk \leftarrow uk^*,\ l_{uk} \leftarrow l_{uk^*},\ S = \{0, j, uk\}$ define

$$F_1(i, j, l, r') \stackrel{\text{def}}{=} g_2^{l\Delta_{j,S}(i) + l_{uk}\Delta_{uk,S}(i)} \left(g_1^{\frac{-f(i)}{i^3+u(i)}} \left(g_2^{i^3+u(i)} g^{f(i)} \right)^{r'} \right)^{\Delta_{0,S}(i)}$$

$$F_2(i, r') \stackrel{\text{def}}{=} \left(g_1^{\frac{-1}{i^3+u(i)}} g^{r'} \right)^{\Delta_{0,S}(i)} .$$

$\mathcal{SK}'(\omega)$:
If $rev = 0$ and $\omega = \omega^*$ then abort
If $rev = 1$ and $\omega = \omega^*$ then
$v \leftarrow v^*$,
$\forall x \in \mathsf{Path}(v)\ r_x \xleftarrow{\$} \mathbb{Z}_p$

if $\nexists l_x$ then $l_x \xleftarrow{\$} \mathbb{Z}_p$ and store l_x in node x

$D_x \leftarrow g_2^{l_x} H_{g_2,h_1,h_2,h_3,h_4}(\omega^*)^{r_x},\ d_x \leftarrow g^{r_x}$

If $rev = 0$ and $\omega \neq \omega^*$ then

Pick an unassigned leaf node v from T and store ω in node v

$\forall x \in \mathsf{Path}(v)\ r'_x \xleftarrow{\$} \mathbb{Z}_p$

if $\nexists l_x$ then $l_x \xleftarrow{\$} \mathbb{Z}_p$ and store l_x in node x

$i \leftarrow \omega, j \leftarrow t^*, l \leftarrow l_x, r' \leftarrow r'_x$

$D_x \leftarrow F_1(i,j,l,r'),\ d_x \leftarrow F_2(i,r')$

If $rev = 1$ and $\omega \neq \omega^*$ then:

Pick an unassigned leaf node v from T and store ω in node v

$\forall x \in \mathsf{Path}(v)\ r'_x \xleftarrow{\$} \mathbb{Z}_p$

if $\nexists l_x$ then $l_x \xleftarrow{\$} \mathbb{Z}_p$ and store l_x in node x

$i \leftarrow \omega, l \leftarrow l_x, r' \leftarrow r'_x$

$\forall x \in (\mathsf{Path}(v) \setminus \mathsf{Path}(v^*))$

$j \leftarrow t^*$

$D_x \leftarrow F_1(i,j,l,r'),\ d_x \leftarrow F_2(i,r')$

$\forall x \in (\mathsf{Path}(v) \cap \mathsf{Path}(v^*))$

$j \leftarrow \omega^*$

$D_x \leftarrow F_1(i,j,l,r'),\ d_x \leftarrow F_2(i,r')$

Return $sk_\omega = \{(x, D_x, d_x)\}_{x \in \mathsf{Path}(v)}$

$\mathcal{R}(\omega, t)$:

For all leaf nodes v associated with identity ω, add (v,t) to rl

$\mathcal{KU}'(t)$:

If $rev = 1, t = t^*$ and $\forall t \leq t^*\ (\omega^*, t) \notin rl$ then abort

Else if $t = t^*$ then:

$\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t), r_x \xleftarrow{\$} \mathbb{Z}_p$

$E_x \leftarrow g_2^{l_x} H_{g_2,h_1,h_2,h_3,h_4}(t^*)^{r_x},\ e_x \leftarrow g^{r_x}$

If $rev = 1, t \neq t^*$ then

$\forall x \in (\mathsf{KUNodes}(\mathsf{T}, rl, t) \setminus \mathsf{Path}(v^*))$

$r'_x \xleftarrow{\$} \mathbb{Z}_p, i \leftarrow t, j \leftarrow t^*, l \leftarrow l_x, r' \leftarrow r'_x$

$E_x \leftarrow F_1(i,j,l,r'),\ e_x \leftarrow F_2(i,r')$

$\forall x \in (\mathsf{KUNodes}(\mathsf{T}, rl, t) \cap \mathsf{Path}(v^*))$

$r'_x \xleftarrow{\$} \mathbb{Z}_p, i \leftarrow t, j \leftarrow \omega^*, l \leftarrow l_x, r' \leftarrow r'_x$

$E_x \leftarrow F_1(i,j,l,r'),\ e_x \leftarrow F_2(i,r')$

If $rev = 0, t \neq t^*$ then

$\forall x \in \mathsf{KUNodes}(\mathsf{T}, rl, t)$

$r'_x \xleftarrow{\$} \mathbb{Z}_p, i \leftarrow t, j \leftarrow t^*, l \leftarrow l_x, r' \leftarrow r'_x$

$E_x \leftarrow F_1(i,j,l,r'),\ e_x \leftarrow F_2(i,r')$

Return $ku_t = \{(x, E_x, e_x)\}_{x \in \mathsf{KUNodes}(\mathsf{T},rl,t)}$

$\mathcal{D}'(\tilde{c})$:

Parse $\tilde{c}$ as $(c = (\omega, t, c_\omega, c_t, c_{uk}, c_1, c_2), \sigma, vk)$

We assume that $\omega = \omega^*, t = t^*$, o.w. such a decryption query can be trivially replied by generating a decryption key using $\mathcal{SK}(\omega)$ and $\mathcal{KU}(t)$ oracles

If $\mathsf{Ver}(vk, c, \sigma) \neq 1$ (i.e. invalid ciphertext), then abort

If $\mathsf{Ver}(vk, c, \sigma) = 1$ and $vk = vk^*$, then return a random bit

If $\mathsf{Ver}(vk, c, \sigma) = 1$ and $vk \neq vk^*$ then

First do a ciphertext sanity check on c (as explained in Section 5)

If c passes sanity check then generate $dk_{\omega,uk} = (D, E, d, e)$ as follows

$uk \leftarrow 01 \| vk,\ r_\omega, l_\omega, l_t, r'_{uk} \xleftarrow{\$} \mathbb{Z}_p,\ S \leftarrow \{0, \omega, t\}$

$D \leftarrow g_2^{l_\omega} H_{g_2,h_1,h_2,h_3,h_4}(\omega)^{r_\omega},\ d \leftarrow g^{r_\omega}$

$$E \leftarrow \left(g_2^{l_\omega \Delta_{\omega,S}(uk) + l_t \Delta_{t,S}(uk)}\right) \cdot \left(g_1^{\frac{-f(uk)}{uk^3+u(uk)}} \left(g_2^{uk^3+u(uk)} g^{f(uk)}\right)^{r'_{uk}}\right)^{\Delta_{0,S}(uk)}$$

$$e \leftarrow \left(g_1^{\frac{-1}{uk^3+u(uk)}} g^{r'_{uk}}\right)^{\Delta_{0,S}(uk)}$$

Now use $dk_{\omega,uk}$ to decrypt c as follows

$$m \leftarrow c_1 \left(\frac{\mathbf{e}(d, c_\omega)}{\mathbf{e}(D, c_2)}\right)^{\frac{uk}{uk-\omega}} \left(\frac{\mathbf{e}(e, c_{uk})}{\mathbf{e}(E, c_2)}\right)^{\frac{\omega}{\omega-uk}}$$

Return m

Analysis. Justification mostly follows directly from the Proof of Theorem 4.3 except for the justification for the simulation of the decryption oracle that we present here. Note that in $\mathcal{RIBE}_{CCA}$, only 2 out of 3 attributes need to be matched between a ciphertext and decryption key for successful decryption, as opposed to 2 out of 2 attributes in $\mathcal{RIBE}$. Therefore, in the decryption algorithm, the ciphertext component corresponding to the verification-key attribute is used only for the ciphertext sanity check and not for actual decryption. However, the decryption oracle needs to use this ciphertext component for decrypting ciphertexts encrypted for the challenge identity and challenge time. The decryption oracle will therefore fail to decrypt malformed ciphertexts, in particular those ciphertexts whose verification key components are invalid. So, the ciphertext sanity check which discards all such ciphertexts with a very high probability, ensures that the decryption oracle will be able to answer all but a negligible fraction of valid decryption queries. Let $q(y)$ be a second-degree polynomial. Let $g_1 = g^a$ (to simplify notation). Define $q(0) = a,\ q(\omega) = l_\omega, q(t) = l_t,\ r_{uk} = (r'_{uk} - \frac{a}{uk^3+u(uk)})\Delta_{0,S}(uk)$ then as in Section 6.3 of [23] we can show that $D = g_2^{q(\omega)} H_{g_2,h_1,h_2,h_3,h_4}(\omega)^{r_\omega},\ d = g^{r_\omega},\ E = g_2^{q(uk)} H_{g_2,h_1,h_2,h_3,h_4}(uk)^{r_{uk}},\ e = g^{r_{uk}}$. From the consistency condition of $\mathcal{RIBE}_{CCA}$ it follows that decryption of c using $dk_{\omega,uk}$ will yield the right message. Hence $\mathcal{B}$ correctly simulates the decryption oracle except for the event when $\mathcal{A}$ queries a ciphertext (c, σ, vk^*) to the decryption oracle where $(c, \sigma) \neq (c^*, \sigma^*)$ and $\mathsf{Ver}(vk^*, c, \sigma) = 1$. We denote this event by forge. Note that the definition of forge is essentially the same as in the Proof of Theorem 1 of [4] but we write it here too for the sake of completeness. Event forge also includes the case where $\mathcal{A}$ queries (c, σ, vk^*) before receiving the challenge ciphetext in which case (c, σ) may or may not be equal to (c^*, σ^*). Now, it is easy to see that (c, σ, vk^*) can be used to forge a signature of $\mathcal{OTS}$ with probability $\Pr[\mathsf{forge}]$. Namely one can construct an adversary F such that $\mathbf{Adv}^{\text{suf-ots}}_{\mathcal{OTS},F}(\kappa) \geq \Pr[\mathsf{forge}]$.

Thus, from Theorem 4.3 and decryption oracle oracle simulation we have

$$\begin{aligned}\mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{RIBE}_{CCA},\mathcal{A},n}(\kappa) &\leq 4 \cdot \mathbf{Adv}^{\text{dbdh}}_{\mathcal{G},\mathcal{B}}(\kappa) + \Pr[\,\mathsf{forge}\,] \\ &\leq 4 \cdot \mathbf{Adv}^{\text{dbdh}}_{\mathcal{G},\mathcal{B}}(\kappa) + \mathbf{Adv}^{\text{suf}-\text{ots}}_{\mathcal{OTS},F}(\kappa)\ .\end{aligned}$$

The above implies the statement of the theorem. ▌

We note that alternatively we could utilize simulation-sound NIZK proofs in a way similar to the construction of CCA secure Fuzzy IBE in [23], but our construction is more efficient.

GENERIC CCA CONSTRUCTION. The Fujisaki-Okamoto (or FO for short) transform [10, 9] is a generic transform to convert a CPA secure public key encryption scheme to a CCA secure one in the RO model. The transform can also be applied to IBE schemes as shown in [16]. Here we show how to apply the FO transform to Revocable IBE schemes. Unlike the previous approach, this solution is generic in that it applies to any Revocable IBE scheme. If applied to our construction (the only secure Revocable IBE scheme currently known), then we suggest to use its more efficient RO modification we discussed, since the FO transform also relies on the RO model.

Let $\mathcal{RIBE} = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ be any Revocable IBE scheme as per Definition 3.1. We can construct another Revocable IBE scheme $\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA} = (\mathcal{S}', \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}', \mathcal{D}', \mathcal{R})$ as follows (we only specify the differences from $\mathcal{RIBE}$). Let $(\mathcal{M}, \mathcal{I}, \mathcal{T})$ and $(\mathcal{M}', \mathcal{I}, \mathcal{T})$ be the (message space, identity space, time space) of $\mathcal{RIBE}$ and $\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA}$ resp. Let $\mathcal{COINS}$ be the set from where $\mathcal{E}$ draws its random coins. We require that for every $m \in \mathcal{M}', \sigma \in \{0,1\}^\kappa$ we have that $m\|\sigma \in \mathcal{M}$. To make the use of randomness explicit we use notation $r \xleftarrow{\$} \mathcal{COINS}\ ;\ \mathcal{E}(\cdot,\cdot,\cdot,\cdot\ ;\ r)$ as opposed to the traditional shorthand $\mathcal{E}(\cdot,\cdot,\cdot,\cdot)$. The setup algorithm $\mathcal{S}'(1^\kappa, n)$ follows $\mathcal{S}$. In addition, it specifies a hash function $\mathcal{H} : \{0,1\}^* \to \mathcal{COINS}$ and outputs it as part of public parameters pk'. The encryption and decryption algorithms are as follows.

- **Encryption** $\mathcal{E}'(pk', \omega, t, m)$:
  $\sigma \xleftarrow{\$} \{0,1\}^\kappa\ ;\ r \leftarrow \mathcal{H}(m\|\sigma\|\omega\|t)$
  $c \leftarrow \mathcal{E}(pk, \omega, t, m\|\sigma\ ;\ r)$
  Return c.

- **Decryption** $\mathcal{D}'(dk_{\omega,t}, c)$:
  $m' \leftarrow \mathcal{D}(dk_{\omega,t}, c)$
  Parse m' as $m\|\sigma\ ;\ r \leftarrow \mathcal{H}(m\|\sigma\|\omega\|t)$
  If $c = \mathcal{E}(pk, \omega, t, m\|\sigma\ ;\ r)$ then return m else return $\perp$.

Consistency follows from the justification of the consistency requirement for $\mathcal{RIBE}$. ▌

Before we present the security analysis of $\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA}$, we define a property γ-uniformity for Revocable IBE schemes.

γ-UNIFORMITY. The γ-uniformity property has been defined earlier in the context of public key encryption schemes [10] and identity-based encryption schemes [26]. Here we define it for Revocable IBE schemes.

Definition 5.2 [**γ-uniformity**] Let $\mathcal{RIBE} = (\mathcal{S}, \mathcal{SK}, \mathcal{KU}, \mathcal{DK}, \mathcal{E}, \mathcal{D}, \mathcal{R})$ be a Revocable IBE scheme. For any public parameter pk output by $\mathcal{S}$, any given identity $\omega \in \mathcal{I}$, time $t \in \mathcal{T}$, the corresponding decryption key dk, message $m \in \mathcal{M}$ and a ciphertext c output by $\mathcal{E}$ define

$$\gamma(m,c) = \Pr\left[\, r \xleftarrow{\$} \mathcal{COINS} : c = \mathcal{E}(pk,\omega,t,m;r) \,\right].$$

We say that $\mathcal{RIBE}$ is γ-uniform if for any $\omega \in \mathcal{I}, t \in \mathcal{T}, m \in \mathcal{M}$ and any ciphertext c output by $\mathcal{E}$, $\gamma(m,c) \leq \gamma$.

$\mathcal{FO}$-$\mathcal{RIBE}_{CCA}$ Security.

Theorem 5.3 Let $\mathcal{RIBE}$ be a γ-uniform Revocable IBE scheme with message space $\mathcal{M}$, identity space $\mathcal{I}$, time space $\mathcal{T}$ and set of coins $\mathcal{COINS}$ for its encryption algorithm. Let $\mathcal{H}$ be a hash function mapping $\mathcal{M}\times\mathcal{I}\times\mathcal{T}$ to $\mathcal{COINS}$ (modeled as the RO) and $\mathcal{FO}$-$\mathcal{RIBE}_{CCA}$ be the associated Revocable IBE scheme as per construction above. Then for an adversary $\mathcal{A}$ attacking sRID-CCA security of $\mathcal{FO}$-$\mathcal{RIBE}_{CCA}$ with n users, whose running time is $t_{\mathcal{A}}$ and who asks q_d decryption queries and q_h random oracle queries, there exists an adversary $\mathcal{B}$ attacking sRID-CPA security of $\mathcal{RIBE}$ such that

$$\mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA},\mathcal{A},n}(\kappa) \leq \left(\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{B},n}(\kappa) + \frac{1}{2}\right)\cdot\left(\frac{1}{1-\gamma q_d}\right) + \frac{q_h}{2^\kappa} - \frac{1}{2}\,, \tag{3}$$

where the running time of $\mathcal{B}$, $t_{\mathcal{B}} \approx t_{\mathcal{A}} + \kappa q_h$.

Proof: We construct an adversary $\mathcal{B}$ attacking sRID-CPA security of $\mathcal{RIBE}$ using adversary $\mathcal{A}$ attacking sRID-CCA security of $\mathcal{FO}$-$\mathcal{RIBE}_{CCA}$ (in RO model). It has access to $\mathcal{SK}(\cdot), \mathcal{KU}(\cdot)$ and $\mathcal{R}(\cdot,\cdot)$ oracles. It needs to simulate random oracle $\mathcal{RO}(\cdot)$ (for hash function $\mathcal{H}$) and decryption oracle $\mathcal{D}'(\cdot)$.

$\mathcal{B}^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot)}(1^\kappa)$:

$(\omega^*, t^*, state) \xleftarrow{\$} \mathcal{A}(1^\kappa)$
Return $(\omega^*, t^*, state)$ and get back pk
Run $\mathcal{A}$ and answer its queries to the $\mathcal{SK}(\cdot), \mathcal{KU}(\cdot)$ and $\mathcal{R}(\cdot,\cdot)$ oracles by using own oracles. For $\mathcal{RO}(\cdot)$ and $\mathcal{D}'(\cdot)$ oracles use subroutines defined below
$(m_0, m_1, state') \xleftarrow{\$} \mathcal{A}^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot),\mathcal{D}'(\cdot),\mathcal{RO}(\cdot)}(1^\kappa, pk, state)$
$\sigma_0, \sigma_1 \xleftarrow{\$} \{0,1\}^\kappa$
Return $(m_0\|\sigma_0, m_1\|\sigma_1, state')$ and get back c^*
$d \xleftarrow{\$} \mathcal{A}^{\mathcal{SK}(\cdot),\mathcal{KU}(\cdot),\mathcal{R}(\cdot,\cdot),\mathcal{D}'(\cdot),\mathcal{RO}(\cdot)}(1^\kappa, pk, c^*, state')$
If any of the oracles or $\mathcal{A}$ aborts then for $i \in \{0,1\}$ find a tuple $(m_i\|\sigma_i, \omega, t, r)$ in the list stored by $\mathcal{RO}$. If such a tuple is found return i else return 1
Else if neither the oracles nor $\mathcal{A}$ aborts return d

Below we present the subroutines for answering $\mathcal{RO}(\cdot)$ and $\mathcal{D}'(\cdot)$ oracle queries.

$\mathcal{RO}(m\|\omega\|t)$:

If (m,ω,t) was queried before then return the corresponding r
Else $r \xleftarrow{\$} \mathcal{COINS}$, store (m,ω,t,r) and return r

$\mathcal{D}'(c)$:
Compute ω, t from c
Find a tuple (m, ω, t, r) stored by $\mathcal{RO}(\cdot)$ oracle, such that $c = \mathcal{E}(pk, \omega, t, m\ ;\ r)$
If no such pair found then abort
Else parse m as $m' \| \sigma$ where $m' \in \mathcal{M}'$ and $\sigma \in \{0,1\}^\kappa$
Return m'

Analysis. Let the challenge bit for the adversary $\mathcal{B}$ be b. Define the following events.

abort: Decryption oracle aborts in the sRID-CCA experiment simulated by $\mathcal{B}$.

succA: $\mathcal{A}$ succeeds in the sRID-CCA experiment simulated by $\mathcal{B}$, given that abort does not occur.

succB: $\mathcal{B}$ succeeds in the sRID-CPA experiment, given that abort does not occur.

queryb: $\mathcal{A}$ queries $(m_b \| \sigma_b, \omega, t, r)$ to the random oracle in the sRID-CCA experiment simulated by $\mathcal{B}$.

queryb$'$: $\mathcal{A}$ queries $(m_{\bar{b}} \| \sigma_{\bar{b}}, \omega, t, r)$ to the random oracle in the sRID-CCA experiment simulated by $\mathcal{B}$.

$$\begin{aligned}
\Pr[\,\mathsf{succA}\,] &= \Pr[\,\mathsf{succA} \mid \mathsf{queryb}\,]\Pr[\,\mathsf{queryb}\,] \\
&\quad + \Pr\left[\,\mathsf{succA} \mid \neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right]\Pr\left[\,\neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right] \\
&\quad + \Pr\left[\,\mathsf{succA} \mid \neg\mathsf{queryb} \wedge \neg\mathsf{queryb}'\,\right]\Pr\left[\,\neg\mathsf{queryb} \wedge \neg\mathsf{queryb}'\,\right]
\end{aligned}$$

$$\begin{aligned}
\Pr[\,\mathsf{succB}\,] &= \Pr[\,\mathsf{succB} \mid \mathsf{queryb}\,]\Pr[\,\mathsf{queryb}\,] \\
&\quad + \Pr\left[\,\mathsf{succB} \mid \neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right]\Pr\left[\,\neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right] \\
&\quad + \Pr\left[\,\mathsf{succB} \mid \neg\mathsf{queryb} \wedge \neg\mathsf{queryb}'\,\right]\Pr\left[\,\neg\mathsf{queryb} \wedge \neg\mathsf{queryb}'\,\right] .
\end{aligned}$$

From the description of $\mathcal{B}$ we have

$$\begin{aligned}
&\Pr[\,\mathsf{succB} \mid \mathsf{queryb}\,] = 1 , \\
&\Pr\left[\,\mathsf{succB} \mid \mathsf{queryb}'\,\right] = 0 , \\
&\Pr\left[\,\mathsf{succA} \mid \neg\mathsf{queryb} \wedge \neg\mathsf{queryb}'\,\right] = \Pr\left[\,\mathsf{succB} \mid \neg\mathsf{queryb} \wedge \neg\mathsf{queryb}'\,\right] .
\end{aligned}$$

Thus we have

$$\begin{aligned}
\Pr[\,\mathsf{succB}\,] - \Pr[\,\mathsf{succA}\,] &= (1 - \Pr[\,\mathsf{succA} \mid \mathsf{queryb}\,])\Pr[\,\mathsf{queryb}\,] \\
&\quad - \Pr\left[\,\mathsf{succA} \mid \neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right]\Pr\left[\,\neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right] \\
&\geq - \Pr\left[\,\neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right] .
\end{aligned}$$

But, we know that the probability that $\mathcal{A}$ queries a $(m_{\bar{b}} \| \sigma_{\bar{b}}, \omega, t, r)$ tuple to the random oracle is at most $2^{-\kappa}$, because $\sigma_{\bar{b}}$ is a randomly chosen κ-bit value that is information theoretically hidden from $\mathcal{A}$. So,

$$\Pr\left[\,\neg\mathsf{queryb} \wedge \mathsf{queryb}'\,\right] \leq \frac{q_h}{2^\kappa}.$$

Also, $\Pr[\,\mathsf{succA}\,] = \mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA},\mathcal{A},n}(\kappa) + 1/2$. So,

$$\Pr[\,\mathsf{succB}\,] \geq \mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA},\mathcal{A},n}(\kappa) + \frac{1}{2} - \frac{q_h}{2^\kappa}\ .$$

It remains to estimate $\Pr[\,\neg\mathsf{abort}\,]$. The event abort occurs when $\mathcal{A}$ queries the decryption oracle for some ciphertext without querying the random oracle for the randomness that was used to generate the ciphertext. From the definition of γ-uniformity Definition 5.2, we know that this can happen with probability at most γ. Thus,

$$\Pr[\,\neg\mathsf{abort}\,] \leq (1-\gamma)^{q_d} \approx (1-\gamma q_d)\ .$$

Therefore,

$$\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{B},n}(\kappa) \geq (1-\gamma q_d)\left(\mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA},\mathcal{A},n}(\kappa) + \frac{1}{2} - \frac{q_{ro}}{2^\kappa}\right) - \frac{1}{2}\ .$$

Thus, Equation (3) follows from the above,

$$\mathbf{Adv}^{\text{srid}-\text{cca}}_{\mathcal{FO}\text{-}\mathcal{RIBE}_{CCA},\mathcal{A},n}(\kappa) \leq \left(\mathbf{Adv}^{\text{srid}-\text{cpa}}_{\mathcal{RIBE},\mathcal{B},n}(\kappa) + \frac{1}{2}\right)\left(\frac{1}{1-\gamma q_d}\right) + \frac{q_{ro}}{2^\kappa} - \frac{1}{2}\ .$$

The above implies the statement of the theorem. ∎

6 Revocable ABE and Fuzzy IBE

Key-policy attribute-based encryption (KP-ABE) [14] is a generalization of Fuzzy IBE which allows the authority to specify more advanced decryption policies. In KP-ABE, as in Fuzzy IBE, each ciphertext is labeled by the sender with a set of descriptive attributes. However, each private key is associated with an access tree that specifies which type of ciphertexts the key can decrypt. A particular key can decrypt a particular ciphertext only if the ciphertext attributes satisfy the access tree of the key. The problem of revocation of attributes is as relevant to KP-ABE as the problem of identity revocation is relevant for IBE. There is no solution known other than the frequent key update for all attributes. As we explained in the Introduction this solution does not scale well. We extend our ideas to construct a *key-policy attribute-based encryption with efficient revocation* or simply *Revocable KP-ABE*. Here we just explain how we obtain a Revocable KP-ABE and that will imply a Revocable Fuzzy IBE as well. The security of our construction holds only in a weaker selective revocation list model, where the adversary must declare in advance all the users to be revoked prior to the challenge time.

The construction uses the KP-ABE construction from [14] and a binary tree in the following way. Messages are encrypted with attributes β and time, where β is the set of attributes which is used in encryption in KP-ABE. The root node of the access tree of decryption key is a 2-out-of-2 gate whose one child is time (similarly to Revocable IBE) and the other child is the root node of access tree $\mathbb{A}$. The component of decryption key corresponding to $\mathbb{A}$ and time are called private key and key update, respectively. Private key for access tree $\mathbb{A}$ is computed in the same way as keys are computed in KP-ABE except that, instead of the root polynomial of $\mathbb{A}$, the root polynomial of decryption key evaluates to the master key at 0. The use of binary tree is essentially the same in both Revocable IBE and Revocable KP-ABE e.g., the way users are assigned to leaf nodes, the way polynomials are selected for each node, the number of private keys each user gets, the way key updates are computed etc.

7 Conclusions

We proposed an IBE scheme with efficient revocation, whose complexity of key updates is significantly reduced (from linear to logarithmic in the number of users) compared to the previous solution. We discussed several variants achieving different levels of security. We also discussed how to construct an attribute-based encryption scheme with efficient revocation. Our schemes should be particularly useful in the settings where a large number of users is involved and scalability is an issue.

8 Acknowledgements

We thank Amit Sahai and Hakan Seyalioglu for pointing out that the security of our revocable KP-ABE holds in a weaker (selective revocation list) model than was claimed in the conference proceedings version. We also thank Adam O'Neill and anonymous reviewers for useful comments and suggestions, and Goichiro Hanaoka for clarifications on [15].

References

[1] W. Aiello, S. Lodha, and R. Ostrovsky. Fast Digital Identity Revocation (Extended Abstract). In *CRYPTO*, pages 137–152, 1998.

[2] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[3] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *EUROCRYPT*, pages 223–238, 2004.

[4] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2006.

[5] D. Boneh, X. Ding, G. Tsudik, and M. Wong. A method for fast revocation of public key certificates and security capabilities. In *USENIX Security Symposium*, pages 22–22, 2001.

[6] D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO*, pages 213–229, 2001.

[7] R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *EUROCRYPT*, pages 207–222, 2004.

[8] Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. In *EUROCRYPT*, pages 255–271, 2003.

[9] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *Public Key Cryptography*, pages 53–68, 1999.

[10] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO*, pages 537–554, 1999.

[11] Craig Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *EUROCRYPT*, pages 272–293, 2003.

[12] V. Goyal. Certificate Revocation Using Fine Grained Certificate Space Partitioning. In *Financial Cryptography*, pages 247–259. Springer, 2007.

[13] V. Goyal. Reducing Trust in the PKG in Identity Based Cryptosystems. In *CRYPTO*, pages 430–447, 2007.

[14] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[15] Y. Hanaoka, G. Hanaoka, J. Shikata, and H. Imai. Identity-Based Hierarchical Strongly Key-Insulated Encryption and Its Application. In *ASIACRYPT*, pages 495–514, 2005.

[16] T. Kitagawa, P. Yang, G. Hanaoka, R. Zhang, H. Watanabe, K. Matsuura, and H. Imai. Generic Transforms to Acquire CCA-Security for Identity Based Encryption: The Cases of FOpkc and REACT. In *ACISP*, pages 348–359, 2006.

[17] B. Libert and J.-J. Quisquater. Efficient revocation and threshold pairing based cryptosystems. In *PODC*, pages 163–171, 2003.

[18] S. Micali. Efficient certificate revocation. *Technical Report MIT/LCS/TM-542b*, 1996.

[19] S. Micali. Novomodo: Scalable Certificate Validation and Simplified PKI Management. In *PKI Research Workshop*, 2002.

[20] D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. In *CRYPTO*, 2002.

[21] M. Naor and K. Nissim. Certificate Revocation and Certificate Update. In *USENIX Security Symposium*, 1998.

[22] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *ACM Conference on Computer and Communications Security*, pages 99–112, 2006.

[23] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT*, pages 457–473, 2005.

[24] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*, pages 47–53, 1984.

[25] B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *EUROCRYPT*, pages 114–127, 2005.

[26] P. Yang, T. Kitagawa, G. Hanaoka, R. Zhang, K. Matsuura, and H. Imai. Applying Fujisaki-Okamoto to Identity-Based Encryption. In *AAECC*, pages 183–192, 2006.