Traceable and Retrievable Identity-Based Encryption*

Man Ho Au[1], Qiong Huang[2], Joseph K. Liu[3], Willy Susilo[1], Duncan S. Wong[2], and Guomin Yang[2]

[1] Centre for Computer and Information Security Research (CCISR)
School of Computer Science and Software Engineering,
University of Wollongong, Australia
{mhaa456, wsusilo}@uow.edu.au
[2] Department of Computer Science, City University of Hong Kong, Hong Kong
csqhuang@student.cityu.edu.hk,
{duncan, csyanggm}@cs.cityu.edu.hk
[3] Cryptography and Security Department
Institute for Infocomm Research, Singapore
ksliu@i2r.a-star.edu.sg

Abstract. Very recently, the concept of Traceable Identity-based Encryption (IBE) scheme (or Accountable Authority Identity based Encryption scheme) was introduced in Crypto 2007. This concept enables some mechanisms to reduce the trust of a private key generator (PKG) in an IBE system. The aim of this paper is threefold. First, we discuss some subtleties in the first traceable IBE scheme in the Crypto 2007 paper. Second, we present an extension to this work by having the PKG's master secret key retrieved automatically if more than one user secret key are released. This way, the user can produce a concrete *proof of misbehaviour* of the PKG in the court. In contrast to previous approach, our idea gives strong incentive for the PKG to strengthen the security of the system since if someone can successfully release a user's secret key, it means that his security is also compromised. We present a formal model to capture our idea. Third, we present an efficient construction based on Gentry's IBE that satisfies our model and prove its security. Our construction is proven secure in the random oracle model. Nevertheless, we should emphasize that the aim of this paper is to introduce the new model to strengthen the IBE system.

Keywords: Identity-based Encryption, Traceability, Retrievability, PKG, Trust.

1 Introduction

The idea of identity-based encryption (IBE) was put forward by Shamir in his seminal paper in [1]. The main concept was proposed in 1984, whilst the first

* M. H. Au and W. Susilo's work was partially supported by ARC Linkage Project LP0667899 and ARC Discovery Grants DP0663306 and DP0877123. Q. Huang, D. S. Wong and G. Yang's work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (RGC Ref. No. CityU 122107).

practical and fully functional IBE scheme was only proposed in [2] that takes advantage of the properties of suitable bilinear maps (the Weil or Tate pairing) over supersingular elliptic curves. Since then, many new schemes have been proposed in the literature (including the recent one by Gentry [3]). The main essence of IBE is to remove the necessity of public key certification, that is required in the conventional public key cryptography setting. The public key of each participant is obtained from his/her public identity, such as email address, IP address combined with a user name, social security number, etc. that can uniquely identify the participant. Furthermore, the sender can send his ciphertext to a recipient without requiring the receiver to have his public key setup first. Indeed, the secret key can be retrieved later after the receiver receives the ciphertext sent by the sender. Unfortunately, this model requires the existence of a *trusted* authority called the Private Key Generator (PKG), whose task is to generate user's private key from their identity information, after a successful identification.

In an IBE system, the PKG is completely trusted, and therefore the PKG has the liberty to engage in any malicious activity without any risk of being sent to court. To mitigate this trust problem, a distributed PKG was proposed [2]. Very recently, Goyal [4] presented a new idea to reduce the trust of the PKG. His idea is to produce an exponential (or super-polynomial) number of possible decryption keys corresponding to every identity. The PKG does not know which secret key that has been chosen by the user. Therefore, when the PKG releases one of the possible user's secret keys, then the user can later show two different secret keys as his proof of the PKG misbehaviour. Goyal formalized this notion as a *traceable identity-based encryption* scheme [4] (This notion was renamed to *Accountable Authority Identity-based Encryption* (A-IBE) in [5]).

Nonetheless, we believe that traceable IBE is *not* very useful for achieving the purpose of deterring the PKG from distributing private keys for any identity. The reason is that in practice, it is difficult for a user to win a court if the user sues the PKG. This is because the PKG can always put a disclaimer well in advance for mitigating the liability of the PKG. Another reason is that the damage is externality with respect to the PKG, rather than the PKG itself. Therefore, there is no strong incentive for the PKG to secure its own system. We therefore motivate ourselves with an additional mechanism which can help discourage the PKG from distributing private keys for any identity while encouraging the PKG to strengthen the security of its own system.

Our Contributions

We take one step forward than Goyal's idea in reducing the trust on the PKG. Our idea is to have the PKG's master secret key retrieved automatically if more than one user secret key are released. This way, the user can produce a concrete *proof of misbehaviour* of the PKG in the court. In contrast to Goyal's approach, our idea also gives some benefit to the PKG to strengthen their security system as if someone can successfully release a user's secret key, it means that his security is also compromised. Therefore, it is also the PKG's interest to ensure its security of the system (c.f. Goyal's approach [4]).

In this paper, firstly we point out some subtleties in Goyal's work [4, 5]. More specifically, there are some subtleties in instantiating the ZK-POK sub-protocol in the first traceable IBE scheme provided in [4, 5] and we propose a suggestion on how to efficiently instantiate it. Furthermore, we observe that several definitions used in Goyal's work are *not* formally defined. Hence, the model provided in Goyal's work is incomplete. We present a formal model to capture our idea mentioned above with the aim of reducing the trust on the PKG. We deal with this part in two stages. Firstly, we formally define the parts that are lacking from Goyal's work. Then, we add an algorithm called `Retrieve` that is used to output the master secret key given two different user's secret keys. We call this notion as a *Traceable and Retrievable Identity-based Encryption scheme*. Second, we present an efficient construction based on Gentry's IBE [3] that satisfies our model.

Paper Organization
The rest of this paper is organized as follows. In Sec. 3, we present some security remarks on Goyal's work [4, 5]. In Sec. 4, we present the formal model of Traceable and Retrievable IBE. We present a concrete construction based on Gentry's IBE in Sec. 5. Finally, we conclude the paper in Sec. 6.

2 Preliminaries

Notations
Let e be a bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of prime order p.
- each element of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ has unique binary representation.
- g, h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- (Bilinear) $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and a, $b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.
- (Non-degenerate) $e(g, h) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be the same or different groups. We say that two groups ($\mathbb{G}_1$, $\mathbb{G}_2$) are a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$ and the bilinear mapping e are all efficiently computable.

Complexity Assumptions
The security of our concrete construction is based on a complexity assumption called "truncated decision q-ABDHE assumption" proposed in [3] which is defined as follows:

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of large prime order p. Given a vector of $q + 3$ elements:

$$\left(g', g'^{(\alpha)^{q+2}}, g, g^{\alpha}, g^{(\alpha)^2}, \ldots, g^{(\alpha)^q}\right) \in \mathbb{G}^{q+3}$$

and an element $Z \in \mathbb{G}_T$ as input, output 0 if $Z = e(g^{(\alpha)^{q+1}}, g')$ and output 1 otherwise.

An algorithm $\mathcal{B}$ has advantage ϵ in solving the truncated decision q-ABDHE if:

$$\Big| \Pr[\mathcal{B}(g', g'^{(\alpha)^{q+2}}, g, g^{\alpha}, \ldots, g^{(\alpha)^q}, e(g^{(\alpha)^{q+1}}, g')) = 0] \\ - \Pr[\mathcal{B}(g', g'^{(\alpha)^{q+2}}, g, g^{\alpha}, \ldots, g^{(\alpha)^q}, Z) = 0] \Big| \geq \epsilon$$

where the probability is over the random choice of generators g, g' in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p$, the random choice of Z in $\mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$.

The computational version of the assumption is defined in the natural way, where the term Z is asked as the output.

Definition 1. *We say that the truncated decision (t, ϵ, q)-ABDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the truncated decision q-ABDHE problem in $\mathbb{G}$.*

3 On Goyal's Scheme [4]

In this section we analyze some subtleties in Goyal's paper [4]. These subtleties are mainly about the instantiation of ZK-POK sub-protocol and the *FindKey* game. We also make some comments on Goyal's definition of Traceable IBE. First of all, we review Goyal's first traceable IBE scheme below.

3.1 Review of Goyal's First Traceable IBE Scheme

Goyal's first scheme [4] is built on top of Gentry's IBE scheme [3]. The basic cryptosystem (Setup, Encryption and Decryption) are taken from Gentry's scheme [3]. The only difference between Goyal's scheme and Gentry's scheme relies on the *Key Generation Protocol*, which is an interactive protocol between a user U and the PKG. For completeness, we review the *Setup* and *Key Generation Protocol* as follows.

Let $\mathbb{G}$ be a bilinear group of large prime order p and let g be a generator of $\mathbb{G}$. Additionally, let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ denote a bilinear map. A security parameter, k, will determine the size of the groups.

Setup. The PKG picks random generators $g, h_1, h_2, h_3 \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It sets $g_1 = g^{\alpha}$ and then selects a hash function H from a family of universal one-way hash function. The published public parameters PK and the master key MK are given by

$$PK = \{g, g_1, h_1, h_2, h_3, H\}, \quad MK = \alpha$$

Key Generation Protocol. This protocol will allow a user U to securely obtain a decryption key d_{ID} from PKG. As in [3], PKG aborts if $\mathsf{ID} = \alpha$. The key generation protocol is as follows.

1. The user U selects a random $r \in \mathbb{Z}_p$ and sends $R = h_1^r$ to the PKG.
2. U gives to PKG a zero-knowledge proof of knowledge of the discrete log of R with respect to h_1.
3. The PKG now selects three random numbers $r', r_{\mathsf{ID},2}, r_{\mathsf{ID},3} \in \mathbb{Z}_p$. It then computes $h'_{\mathsf{ID},1} = (Rg^{-r'})^{1/(\alpha-\mathsf{ID})}$ and $h_{\mathsf{ID},i} = (h_i g^{-r_{ID,i}})^{1/(\alpha-\mathsf{ID})}$, $i \in \{2,3\}$ and sends
$$\{r', h'_{\mathsf{ID},1}, r_{\mathsf{ID},2}, h_{\mathsf{ID},2}, r_{\mathsf{ID},3}, h_{\mathsf{ID},3}\}$$
to the user U.
4. U computes $r_{\mathsf{ID},1} = r'/r$ and $h_{\mathsf{ID},1} = (h'_{\mathsf{ID},1})^{1/r}$. It sets the decryption key $d_{\mathsf{ID}} = \{(r_{\mathsf{ID},i}, h_{\mathsf{ID},i}) : i \in \{1,2,3\}\}$.
5. U now runs a key sanity check on d_{ID} as follows. It computes $g^{\alpha-\mathsf{ID}} = g_1/g^{\mathsf{ID}}$ and checks if $e(h_{\mathsf{ID},i}, g^{\alpha-\mathsf{ID}}) \overset{?}{=} e(h_i g^{-r_{\mathsf{ID},i}}, g)$ for $i \in \{1,2,3\}$. U aborts if the check fails for any i.

At the end of this protocol, U will have a well-formed decryption key d_{ID} for the identity ID.

3.2 Comments on the Instantiation of ZK-POK

In the above scheme, a user runs a *Key Generation Protocol*, which is a zero knowledge proof of knowledge (ZK-POK), with the PKG to jointly generate his/her secret key, without letting the PKG know which key was actually generated. In this protocol, the user first randomly selects R and then proves to the PKG that he/she knows the discrete log of R with respect to base h_1. We believe that the *Key Generation Protocol* is correct. However, there are some subtleties in instantiating the ZK-POK sub-protocol. Goyal suggested to employ Schnorr's 3-round identification scheme [6] as the underlying ZK-POK, which is an *honest-verifier* zero-knowledge proof of knowledge. The revised and extended version [5] does not discuss much about the instantiation either. Proof systems proposed in [7] may not fit the *Key Generation Protocol*, as they merely concentrate on honest verifiers as well. It turns out that efficient instantiations of ZK-POK is not a very trivial task as one may originally think. Below are the subtleties in the instantiations.

In the proof of Theorem 2 in [4], after receiving the challenge R from its challenger, the adversary $\mathcal{B}$, who wishes to solve the discrete log problem, runs the simulator to prove the knowledge of the discrete log of R with respect to base h_1 (by rewinding $\mathcal{A}$ who tries to win the *FindKey* game). Note that in the proof, $\mathcal{B}$ and $\mathcal{A}$ play the roles of the simulator and the verifier, respectively. The simulator $\mathcal{S}$ for Schnorr's protocol does not need to rewind the (honest) verifier in order to provide a transcript indistinguishable from that of a real interaction. It even does not need to interact with the verifier during the simulation at all, since the verifier is assumed to be honest, and $\mathcal{S}$ can select a random challenge on behalf of the verifier. But in the case here, $\mathcal{B}$ has to interact with $\mathcal{A}$ in order to provide an indistinguishable simulation, thus it has to rewind $\mathcal{A}$ to gain some advantage for the simulation. However, there is only one possible way for $\mathcal{B}$ to rewind $\mathcal{A}$. That is, after receving the challenge c from $\mathcal{A}$, $\mathcal{B}$ rewinds $\mathcal{A}$ back to the initial state at the beginning of the zero-knowledge proof, and sends $\mathcal{A}$ a new first-round message a, which is computed based on the challenge c obtained from $\mathcal{A}$. Now, if $\mathcal{A}$ sends back the same c, then the simulation can be completed successfully. However, there is no guarantee on that $\mathcal{A}$ would do so, since $\mathcal{A}$'s status is changed. In a consequence, $\mathcal{B}$ cannot use $\mathcal{S}$ to provide a zero-knowledge proof desired by $\mathcal{A}$ without the knowledge of $\log_{h_1} R$. Hence, $\mathcal{A}$ may not win the *FindKey* game with probability ϵ again.

An oversimplified way to fix the problem is to let verifier V commit itself to its challenge first before prover P sends its first message. P sends its final response only if V reveals the commitment correctly. It is easy to see that the resulting protocol is zero-knowledge against arbitrary verifier, however, it does not seem to be a proof of knowledge.

A better instantiation of the ZK-POK is to use the efficient 6-round ZK-POK of [8] (which can further be compressed into 4 rounds). The verifier V commits to its challenge and proves to P that he knows the challenge. After that P proves to V that he knows either the challenge or the discrete log. Readers can refer to [8] for details.

We emphasize here again that the above comments do not imply that Goyal's scheme is problematic. Instead, they are regarding to efficient instantiations of the ZK-POK sub-protocol.

3.3 Comments on the Definition of [4]

We also notice that the definition for Traceable IBE given by Goyal in [4] is incomplete and imprecise. Comparing with conventional IBE definition [2], a Traceable IBE scheme described in [4] additionally requires the user to do a "sanity check" on the "well-formedness" of an extracted user secret key from the PKG. However, the meaning of "sanity check" of a user secret key in association with that of "well-formedness" have never been formalized.

Since the PKG is no longer trusted fully in the setting of this research work, the user secret key generated by the PKG using Extract Protocol may be malformed. In this scenario, it is possible that this malformed key can still decrypt a portion of all possible ciphertexts for the user but not all. Now if the malicious PKG publishes another user secret key which can decrypt all the ciphertexts for the user, the user will not be able to win a court if the user provides these two keys as evidence to a court of law, claiming that the PKG is cheating. This is because the PKG can show that one of the keys presented by the user is not a valid key since it cannot decrypt all possible ciphertexts.

Therefore, we believe that the notion of "sanity check" has to be formalized. In addition, in the subsequent security model, we also need to formalize the intuition that a user should be provided with a method to make sure that the user secret key extracted from the PKG via Extract Protocol can always be able to decrypt ciphertexts for the user.

Also due to the lack of "sanity check" definition in [4], the security model of [4] is also incomplete. The attack scenario described above is not captured in any of the models specified in [4].

4 Traceable and Retrievable IBE

4.1 TR-IBE Model

A Traceable and Retrievable Identity-Based Encryption (TR-IBE) scheme consists of six probabilistic polynomial-time (PPT) algorithms and one two-party interactive protocol.

Setup. On input 1^k where $k \in \mathbb{N}$ is a security parameter, it outputs a master public/secret key pair (mpk, msk).

$\mathsf{Extract\ Protocol}$. The Private Key Generator, PKG, on input (mpk, msk, ID) carries out the protocol with a user on input (mpk, ID). At the end of the protocol, the user outputs a user secret key usk_{ID} or a symbol $\perp$ indicating the failure of the protocol run. Formally, the PKG and the user in the protocol are considered as PPT Turing machines. Without loss of generality, we let $\mathsf{ID} \in \{0,1\}^k$. In practice, the user with identity ID may send the identity to the PKG in the first message flow of the protocol.

$\mathsf{SanityCheck}$. On input $(mpk, \mathsf{ID}, usk_{\mathsf{ID}})$, it outputs 1 or 0.

Enc. On input (mpk, ID, m), where m is a message from a message space $\mathcal{M}$ defined by mpk, it outputs a ciphertext C.

Dec. On input $(mpk, usk_{\mathsf{ID}}, C)$, it outputs $m \in \mathcal{M}$ or a symbol $\perp$ if the decryption fails.

Trace[1]. On input $(mpk, \mathsf{ID}, usk_{\mathsf{ID}})$, it outputs $\perp$ if $\mathsf{SanityCheck}(mpk, \mathsf{ID}, usk_{\mathsf{ID}}) \neq 1$. Otherwise it outputs a user key family number fn_{ID} from a user key family number space denoted by $\mathcal{F}_{\mathsf{ID}}$. This space is defined by (mpk, ID).

$\mathsf{Retrieve}$. On input $(mpk, \mathsf{ID}, usk_{\mathsf{ID}}, \widetilde{usk}_{\mathsf{ID}})$, it outputs the master secret key msk or a symbol $\perp$ indicating the failure of retrieval.

For correctness, we require that for all $k \in \mathbb{N}$, for any $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$, any identity $\mathsf{ID} \in \{0,1\}^k$, any $usk_{\mathsf{ID}} \neq \perp$ output by the user with identity ID at the end of a run of $\mathsf{Extract\ Protocol}$ with the PKG, any $m \in \mathcal{M}(mpk)$, we have

1. $1 \leftarrow \mathsf{SanityCheck}(mpk, \mathsf{ID}, usk_{\mathsf{ID}})$;
2. $m \leftarrow \mathsf{Dec}(mpk, usk_{\mathsf{ID}}, \mathsf{Enc}(mpk, \mathsf{ID}, m))$; and
3. $\mathsf{Trace}(mpk, \mathsf{ID}, usk_{\mathsf{ID}}) \in \mathcal{F}_{\mathsf{ID}}$.

4.2 Security Model for TR-IBE

In [4], three games have been given for formalizing the following three security notions.

1. Confidentiality of Ciphertexts: a conventional indistinguishability based game capturing chosen ciphertext and identity attacks, namely $\mathsf{IND\text{-}ID\text{-}CCA}$ similar to that in [2] is given.

[1] Trace is needed for some technical reason. It is used for formalizing the user key family number and will mainly be used in the $\mathsf{Retrievability\ Game}$ described on page 103.

2. Secrecy of User Secret Key Against Malicious PKG: the PKG, after carrying out a successful run of Extract Protocol with a user, should not be able to find out the user key family number of the user secret key obtained by the user.
3. Security Against Framing by Malicious Users: a user should not be able to come up with two user secret keys such that the corresponding user key family numbers are different, after at most one execution of Extract Protocol with the PKG.

As explained in Sec. 3.3, an additional security notion related to "sanity check" of user secret key should be introduced. We also introduce the security notion "Retrievability" which is specific to the retrievability property of TR-IBE. It can be seen that the security against framing by malicious user in TR-IBE is implied by the requirement of confidentiality of ciphertexts and the retrievability property. If the adversary is able to come up with two user secret keys, he can compute the master secret key of PKG due to the retrievability property and is able to break the confidentiality of ciphertexts of all users. In the following, we formalize all these security notions. The corresponding games are Confidentiality (IND-ID-CCA) Game, FindKey Game, SanityCheck Game and Retrievability Game.

Confidentiality (IND-ID-CCA) Game. On input a security parameter 1^k, $k \in \mathbb{N}$, the following game is carried out by a simulator $\mathcal{S}$ against an adversary $\mathcal{A}$.

1. $\mathcal{S}$ generates $(mpk, msk) \leftarrow \mathsf{Setup}(1^k)$ and invokes $\mathcal{A}$ on mpk. $\mathcal{S}$ maintains a list L. Initially, L is set to empty.
2. $\mathcal{A}$ (which acts as a user) may execute Extract Protocol with $\mathcal{S}$ (which acts as the PKG) on any identity ID or query a decryption oracle ODec on (ID, C). For each run of Extract Protocol, if $\mathsf{ID} \in L$, $\mathcal{S}$ rejects the protocol run immediately[2]. Otherwise, the protocol is carried out. At the end of a successful run, $\mathcal{S}$ sets $L := L \cup \{\mathsf{ID}\}$. For an ODec query, $\mathcal{S}$ generates a user secret key by simulating Extract Protocol and uses it to decrypt the querying ciphertext.
3. $\mathcal{A}$ submits two equal length messages $m_0^*, m_1^* \in \mathcal{M}(mpk)$ and an identity ID^* to $\mathcal{S}$. If $\mathsf{ID}^* \in L$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ flips a random coin $b \overset{R}{\leftarrow} \{0,1\}$ and sends $C^* \leftarrow \mathsf{Enc}(mpk, \mathsf{ID}^*, m_b^*)$ to $\mathcal{A}$. $\mathcal{S}$ sets $L := L \cup \{\mathsf{ID}\}$.
4. $\mathcal{A}$ can continue executing Extract Protocol and querying ODec. At the end of the game, $\mathcal{A}$ outputs its guess b' of b.

$\mathcal{A}$ wins if $b' = b$ and (ID^*, C^*) has never been queried to ODec. The advantage of $\mathcal{A}$ in this game is defined as $\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$.

In the following, we give the formal definition of "Secrecy of User Secret Key Against Malicious PKG". In Sec. 4.3, we will see that it is stronger than the one given in [4].

FindKey Game. On input a security parameter 1^k, $k \in \mathbb{N}$, the following game is carried out by a simulator $\mathcal{S}$ against an adversary $\mathcal{A}$.

[2] This restriction is natural as if $\mathcal{A}$ were allowed to run Extract Protocol on an ID for multiple times, due to the retrievability property of TR-IBE, msk may be compromised.

1. $\mathcal{S}$ maintains two lists L_1 and L_2, both of them are initialized to null. $\mathcal{S}$ invokes $\mathcal{A}$ on 1^k and gets mpk from $\mathcal{A}$.
2. $\mathcal{A}$ (acts as the malicious PKG) may execute $\mathsf{Extract\ Protocol}$ with $\mathcal{S}$ (which acts as a user) on any identity ID chosen by $\mathcal{A}$. If $\mathsf{ID} \in P(L_1) \cup L_2$, $\mathcal{S}$ rejects the protocol run immediately, where $P(L_1)$ is the collection of the first elements of all the pairs in L_1. At the end of a successful run, suppose the user secret key generated is usk_{ID}. If $\mathsf{SanityCheck}(mpk, \mathsf{ID}, usk_{\mathsf{ID}}) = 1$, $\mathcal{S}$ sets $L_1 := L_1 \cup (\mathsf{ID}, usk_{\mathsf{ID}})$.
3. Since $\mathcal{A}$ (which acts as the malicious PKG) may collude with some users in this multi-user setting (see Sec. 4.3 for more details), $\mathcal{A}$ is allowed to access an oracle called $\mathsf{OCorrupt}$. On input ID, if $(\mathsf{ID}, usk_{\mathsf{ID}}) \in L_1$ for some user secret key usk_{ID}, the oracle returns usk_{ID}, and sets $L_1 := L_1 \setminus \{(\mathsf{ID}, usk_{\mathsf{ID}})\}$ and $L_2 := L_2 \cup \{\mathsf{ID}\}$. Otherwise, $\perp$ is returned.
4. At the end of the game, $\mathcal{A}$ outputs an identity ID^* and a user secret key $\widetilde{usk}_{\mathsf{ID}^*}$.

$\mathcal{A}$ wins if

1. $1 \leftarrow \mathsf{SanityCheck}(mpk, \mathsf{ID}^*, \widetilde{usk}_{\mathsf{ID}^*})$;
2. $(\mathsf{ID}^*, usk_{\mathsf{ID}^*}) \in L_1$; and[3]
3. $\mathsf{Trace}(mpk, \mathsf{ID}^*, usk_{\mathsf{ID}^*}) = \mathsf{Trace}(mpk, \mathsf{ID}^*, \widetilde{usk}_{\mathsf{ID}^*})$.

The advantage of $\mathcal{A}$ in this game is defined as $\Pr[\mathcal{A} \text{ wins}]$.

We now formalize the notion related to "sanity check". As discussed in Sec. 3.3, a user should be provided with a method to make sure that the user secret key extracted from the PKG via $\mathsf{Extract\ Protocol}$ can always be able to decrypt ciphertexts for the user. We consider the following game. Informally, it requires that for any two user secret keys of an identity that passed the "sanity check", both of them will always produce the identical result in decryption.

SanityCheck Game. On input a security parameter 1^k, $k \in \mathbb{N}$, a simulator $\mathcal{S}$ invokes an adversary $\mathcal{A}$ on 1^k. $\mathcal{A}$ returns a master public key mpk, an identity ID^*, two user secret keys usk_{ID^*} and $\widetilde{usk}_{\mathsf{ID}^*}$ and a ciphertext C. $\mathcal{A}$ wins if

1. $1 \leftarrow \mathsf{SanityCheck}(mpk, \mathsf{ID}^*, usk_{\mathsf{ID}^*})$;
2. $1 \leftarrow \mathsf{SanityCheck}(mpk, \mathsf{ID}^*, \widetilde{usk}_{\mathsf{ID}^*})$;
3. $\perp \neq \mathsf{Dec}(mpk, usk_{\mathsf{ID}^*}, C)$;
4. $\perp \neq \mathsf{Dec}(mpk, \widetilde{usk}_{\mathsf{ID}^*}, C)$; and
5. $\mathsf{Dec}(mpk, usk_{\mathsf{ID}^*}, C) \neq \mathsf{Dec}(mpk, \widetilde{usk}_{\mathsf{ID}^*}, C)$.

The advantage of $\mathcal{A}$ in this game is defined as $\Pr[\mathcal{A} \text{ wins}]$. By combining the notion captured in this game and the correctness requirement defined at the beginning of Sec. 4, it is easy to see that the intuition of "sanity check" is captured.

[3] Note that we do not need the restriction that $\mathsf{ID}^* \notin L_2$ as ID^* cannot co-exist in both L_1 and L_2.

The last security notion, also the only one specific to "Retrievability" property, requires that for any two user secret keys corresponding to the same identity but with different user key family numbers, they allow the public to retrieve the master secret key. The following game considers a malicious PKG which tries to come up with two user secret keys such that its master secret key is not retrievable.

Retrievability Game. On input a security parameter 1^k, $k \in \mathbb{N}$, a simulator $\mathcal{S}$ invokes an adversary $\mathcal{A}$ on 1^k. $\mathcal{A}$ returns a master public key mpk, an identity ID^*, and two user secret keys usk_{ID^*} and $\widetilde{usk}_{\mathsf{ID}^*}$. $\mathcal{A}$ wins if

1. $1 \leftarrow \mathsf{SanityCheck}(mpk, \mathsf{ID}^*, usk_{\mathsf{ID}^*})$;
2. $1 \leftarrow \mathsf{SanityCheck}(mpk, \mathsf{ID}^*, \widetilde{usk}_{\mathsf{ID}^*})$;
3. $\mathsf{Trace}(mpk, \mathsf{ID}^*, usk_{\mathsf{ID}^*}) \neq \mathsf{Trace}(mpk, \mathsf{ID}^*, \widetilde{usk}_{\mathsf{ID}^*})$; and
4. $\perp \leftarrow \mathsf{Retrieve}(mpk, \mathsf{ID}^*, usk_{\mathsf{ID}^*}, \widetilde{usk}_{\mathsf{ID}^*})$.

The advantage of $\mathcal{A}$ in this game is defined as $\Pr[\mathcal{A} \text{ wins}]$.

Theorem 1. *A TR-IBE scheme is said to be* secure *if for all polynomial time adversaries, the advantage in each of the* Confidentiality *game,* FindKey *game,* SanityCheck *game and* Retrievability *game is* negligible *in the security parameter k.*

4.3 Further Comments on the Security Model of [4]

The FindKey Game defined in [4] is weaker than our model defined above. In particular, the game in [4] requires the adversary to fix the challenging identity ID^* at the beginning of the game and no further change is allowed. Also, the adversary is not allowed to interact with other identities. In our definition instead, we allow the adversary to "try out" and also corrupt a couple of identities in the manner of *adaptive chosen identity attack* before choosing a challenging identity at the end of the game.

5 A Concrete Scheme

5.1 Construction

High Level Description. Our scheme is based on Gentry's IBE scheme [3]. We extend the scheme by adding a Verifiable Encrypton (VE) scheme [9]. The PKG has two sets of public key pair. One is for the IBE and the other for the VE. At the beginning, the PKG verifibly encrypts, using its VE public key, the (IBE) master secret key. The encrypted master secret key is published. In the key extraction process, one addition component is given to the user as the secret key. This component allows the revocation of the encrypted master secret key. That is, if the PKG generates two secret keys (they may be different) for the same user, by using these two secret keys, the master secret key can be decrypted and revoked.

Technical Details.
Setup: On input 1^k, where $k \in \mathbb{N}$ is a security parameter, let $\mathbb{G}$ and $\mathbb{G}_T$ be groups of order p such that p is a k-bit prime, and let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be the bilinear map. We use a multiplicative notation for the operation in $\mathbb{G}$ and $\mathbb{G}_T$. Let $H_I : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_t : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be secure cryptographic hash functions. The PKG selects four random generators $g, h_1, h_2, h_3 \in \mathbb{G}$ and randomly chooses $\alpha, \beta, x \in_R \mathbb{Z}_p$. It sets $g_1 = g^\alpha, g' = g^\beta$ and $X = e(g,g)^x$. Define the message space $\mathcal{M} = \mathbb{G}_T$.

Then the PKG runs the non-interactive verifiable encryption algorithm from [9]. The idea is to verifiably encrypt the master secret key α and β under the public key X. This can be done as follows. Let $N := \mathrm{poly}(k)$ for some polynomial $\mathrm{poly}(\cdot)$, be a security parameter[4]. Let $H_E : \{0,1\}^* \rightarrow \{0,1\}^N$ be a secure hash function.

1. The PKG randomly selects $u_j, u'_j \in_R \mathbb{Z}_p$ and computes $(T_j = g^{u_j}, T'_j = g^{u'_j})$, for $j = 1$ to N.
2. For $j = 1$ to N the PKG computes the following.
   - Compute $Z_{j,0} = u_j$, $Z_{j,1} = u_j - \alpha$, $Z'_{j,0} = u'_j$, $Z'_{j,1} = u'_j - \beta$.
   - Randomly select $v_{j,0}, v_{j,1}, v'_{j,0}, v'_{j,1} \in_R \mathbb{Z}_p$, compute $E_{0,j,i} = X^{v_{j,i}} \oplus Z_{j,i}, E_{1,j,i} = g^{v_{j,i}}$ and $E'_{0,j,i} = X^{v'_{j,i}} \oplus Z'_{j,i}, E'_{1,j,i} = g^{v'_{j,i}}$ for $i \in \{0,1\}$[5].
3. PKG computes $L = H_E(\ T_1||\ E_{0,1,0}||\ E_{1,1,0}||\ E_{0,1,1}||\ E_{1,1,1}||\ \ldots||\ T_N||\ E_{0,N,0}||\ E_{1,N,0}||\ E_{0,N,1}||\ E_{1,N,1})$ and $L' = H_E(\ T'_1||\ E'_{0,1,0}||\ E'_{1,1,0}||\ E'_{0,1,1}||\ E'_{1,1,1}||\ \ldots||\ T'_N||\ E'_{0,N,0}||\ E'_{1,N,0}||\ E'_{0,N,1}||\ E'_{1,N,1})$. Let b_j, b'_j be the j-th bit of L and L' respectively.
4. Output $\mathcal{T} = \{\ \big(T_j, T'_j, E_{0,j,0}, E_{1,j,0}, E_{0,j,1}, E_{1,j,1}, E'_{0,j,0}, E'_{1,j,0}, E'_{0,j,1}, E'_{1,j,1}, Z_{j,b_j}, v_{j,b_j}, Z'_{j,b'_j}, v'_{j,b'_j}\ \big)\}_{j=1}^N$.
5. Verification. Anyone can check if $\mathcal{T}$ is a valid encryption of α and β under the public key X by computing L and L' from $\mathcal{T}$ and checking if the following equations hold:

$$\begin{aligned} E_{0,j,b_j} &\stackrel{?}{=} X^{v_{j,b_j}} \oplus Z_{j,b_j} \\ E_{1,j,b_j} &\stackrel{?}{=} g^{v_{j,b_j}} \\ T_j &\stackrel{?}{=} g_1^{b_j} g^{Z_{j,b_j}} \\ E'_{0,j,b'_j} &\stackrel{?}{=} X^{v'_{j,b'_j}} \oplus Z'_{j,b'_j} \\ E'_{1,j,b'_j} &\stackrel{?}{=} g^{v'_{j,b'_j}} \\ T'_j &\stackrel{?}{=} g'^{b'_j} g^{Z'_{j,b'_j}} \end{aligned}$$

   where $j = 1$ to N and b_j, b'_j are the j-th bit of L and L' respectively.

The public parameters *mpk* and master secret key *msk* are given by

$$mpk = (g, g_1, g', h_1, h_2, h_3, X, H, H_I, H_t, \mathcal{M}, \mathcal{T}) \qquad msk = (\alpha, \beta, g^x)$$

[4] N controls the cheating probability of the verifiable encryption.
[5] We assume that some appropriate padding has been added for the $\oplus$ operation.

<u>Extract Protocol:</u> On input the master public key/secret key pair (mpk, msk) and an identity $\mathsf{ID} \in \{0,1\}^k$ of a user U, the PKG carries out an interactive protocol with U, as follows. Compute $\mathbb{ID} = H_I(\mathsf{ID})$. If $\mathbb{ID} = \alpha$, it aborts. Otherwise, the protocol proceeds as follow:

1. The user U selects a random $r \in_R \mathbb{Z}_p$ and sends $R = h_1^r$ to the PKG.
2. U gives to the PKG the following zero-knowledge proof of knowledge:
$$PK\{r : R = h_1^r\}$$
3. The PKG randomly selects $r', r_{\mathbb{ID},2}, r_{\mathbb{ID},3} \in \mathbb{Z}_p$ and computes
$$h'_{\mathbb{ID}} = (Rg^{-r'})^{1/(\alpha - \mathbb{ID})} \qquad t_{\mathbb{ID}} = g^x h_1^{\frac{H_t(R,r')}{\beta + \mathbb{ID}}}$$
$$h_{\mathbb{ID},2} = (h_2 g^{-r_{\mathbb{ID},2}})^{1/(\alpha - \mathbb{ID})} \qquad h_{\mathbb{ID},3} = (h_3 g^{-r_{\mathbb{ID},3}})^{1/(\alpha - \mathbb{ID})}$$
and sends $(r', h'_{\mathbb{ID}}, t_{\mathbb{ID}}, r_{\mathbb{ID},2}, h_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},3})$ to U.
4. The PKG computes $\pi_{\mathbb{ID}}$, the non-interactive proof statement of the following zero-knowledge proof of knowledge:
$$\pi_{\mathbb{ID}} = PK\Big\{(x, \alpha, \beta) :\ X = e(g,g)^x \ \wedge\ g_1 = g^\alpha \ \wedge\ g' = g^\beta \ \wedge$$
$$h'_{\mathbb{ID}} = R^{\frac{1}{\alpha - \mathbb{ID}}} g^{\frac{-r'}{\alpha - \mathbb{ID}}} \ \wedge\ t_{\mathbb{ID}} = g^x h_1^{\frac{H_t(R,r')}{\beta + \mathbb{ID}}} \Big\}$$
and sends to U.
5. After checking $\pi_{\mathbb{ID}}$, U computes
$$r_{\mathbb{ID},1} = r'/r \qquad h_{\mathbb{ID},1} = (h'_{\mathbb{ID}})^{1/r}$$
The secret key usk_{ID} is $(r, r_{\mathbb{ID},1}, h_{\mathbb{ID},1}, r_{\mathbb{ID},2}, h_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}})$ where $h_{\mathbb{ID},1} = (h_1 g^{-r_{\mathbb{ID},1}})^{1/(\alpha - \mathbb{ID})}$ and $t_{\mathbb{ID}} = g^x h_1^{\frac{H_t(h_1^r, r r_{\mathbb{ID},1})}{\beta + \mathbb{ID}}}$.

<u>SanityCheck:</u> On input (mpk, usk_{ID}) and an identity $\mathsf{ID} \in \{0,1\}^k$, compute $\mathbb{ID} = H_I(\mathsf{ID})$ and check whether $e(h_{\mathbb{ID},i}, g_1/g^{\mathbb{ID}}) = e(h_i g^{-r_{\mathbb{ID},i}}, g)$ for $i = 1,2,3$. Also check whether $\pi_{\mathbb{ID}}$ is a valid proof statement. If all of them are correct, output 1. Otherwise, output 0.

<u>Enc:</u> To encrypt a message $m \in \mathbb{G}_T$ using identity $\mathsf{ID} \in \{0,1\}^k$, compute $\mathbb{ID} = H_I(\mathsf{ID})$, generate a random $s \in_R \mathbb{Z}_p$ and output the ciphertext C where:
$$\begin{aligned} C &= (C_1, C_2, C_3, C_4, C_5) \\ &= \Big(g_1^s g^{-s\mathbb{ID}} \ ,\ e(g,g)^s \ ,\ m \cdot e(g,h_1)^{-s} \ ,\ e(g,h_2)^s e(g,h_3)^{s\gamma}, \pi_C \Big) \end{aligned}$$
where $\gamma = H(C_1, C_2, C_3)$ and π_C is a non-interactive proof statement of the following zero-knowledge proof of knowledge
$$\pi_C = PK\Big\{(s) :\ C_1 = \big(g_1 g^{-\mathbb{ID}}\big)^s \ \wedge\ C_2 = e(g,g)^s \ \Big\}$$

Dec: To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4, C_5)$ using secret key usk_{ID}, compute $\gamma = H(C_1, C_2, C_3)$ and test whether

$$C_4 = e(C_1, h_{\mathbb{ID},2}{h_{\mathbb{ID},3}}^{\gamma}) \cdot {C_2}^{r_{\mathbb{ID},2}+r_{\mathbb{ID},3}\gamma}$$

If it is not equal or π_C is not a valid proof statement , output $\perp$. Else output

$$m = C_3 \cdot e(C_1, h_{\mathbb{ID},1}) \cdot {C_2}^{r_{\mathbb{ID},1}}$$

Trace: On input $(mpk, \mathsf{ID}, usk_{\mathsf{ID}})$, define family space $\mathcal{F}_{\mathsf{ID}} = \mathbb{Z}_p$. Parse $usk_{\mathsf{ID}} = (r, r_{\mathbb{ID},1}, h_{\mathbb{ID},1}, r_{\mathbb{ID},2}, h_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}})$ and output the decryption key family number $f_{n_{\mathbb{ID}}} = r_{\mathbb{ID},1}$.

Retrieve: On input (mpk, ID) and two sets of secret key $(usk_{\mathsf{ID}}, \widetilde{usk}_{\mathsf{ID}})$ for the same identity ID,

1. Compute $\mathbb{ID} = H_I(\mathsf{ID})$ and parse $usk_{\mathsf{ID}} = (r, r_{\mathbb{ID},1}, h_{\mathbb{ID},1}, r_{\mathbb{ID},2}, h_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}})$ and $\widetilde{usk}_{\mathsf{ID}} = (\tilde{r}, \tilde{r}_{\mathbb{ID},1}, \tilde{h}_{\mathbb{ID},1}, \tilde{r}_{\mathbb{ID},2}, \tilde{h}_{\mathbb{ID},2}, \tilde{r}_{\mathbb{ID},3}, \tilde{h}_{\mathbb{ID},3}, \tilde{t}_{\mathbb{ID}}, \tilde{\pi}_{\mathbb{ID}})$
2. Compute $K := H_t(h_1^r, r r_{\mathbb{ID},1})$ and $\tilde{K} := H_t(h_1^{\tilde{r}}, \tilde{r}\tilde{r}_{\mathbb{ID},1})$. If $K = \tilde{K}$, output $\perp$. Otherwise, compute
$$\left(\frac{{t_{\mathbb{ID}}}^{\tilde{K}}}{\tilde{t}_{\mathbb{ID}}^{K}}\right)^{\frac{1}{\tilde{K}-K}} = g^x \tag{1}$$
and check whether $X \overset{?}{=} e(g, g^x)$. If not, output $\perp$.
3. For any $j \in \{1, \dots, N\}$, one can get $\big(T_j, Z_{j,b_j}, E_{0,j,1-b_j}, E_{1,j,1-b_j}\big)$ in $\mathcal{T}$(the verifiable encryption in Setup). For simplicity, we omit the subscript j. That is, one can get $(T := g^u,\ Z_b,\ E_{0,1-b}, E_{1,1-b} := g^{v_{1-b}})$, such that $b \in \{0, 1\}$ where
$$Z_b = u - b\alpha \tag{2}$$
Compute $e(E_{1,1-b}, g^x) \oplus E_{0,1-b}$ to get
$$Z_{1-b} = u - (1-b)\alpha \tag{3}$$
From equation (2) and (3), compute α. Check whether $g_1 \overset{?}{=} g^{\alpha}$. If not, use another $j \in \{1, \dots, N\}$ to compute α. If all j's have been used and no equality is attained, output $\perp$. Otherwise, compute β in the similar way and output (α, β, g^x) as msk.

5.2 Security Analysis

Theorem 2. *The advantage of an adversary in the* IND-ID-CCA *game is negligible for the proposed scheme under the decisional truncated q-ABDHE assumption in the random oracle model.*

Proof: (sketch) The proof in our setting very much falls along the lines of the proof of IND-ID-CCA security of Goyal's scheme [4]. Here we just give a sketch highlighting the differences.

The differences between [4] and our scheme (in terms of IND-ID-CCA) are the formation of public parameter and the generation of a decryption key. In our scheme, $mpk = (g, g_1, g', h_1, h_2, h_3, X, H, H_t, H_I, , \mathcal{M}, \mathcal{T})$ where $(g, g_1, h_1, h_2, h_3, \mathcal{M})$ are generated in the same way as in [4]. The remaining parameters are generated as follows. The simulator $\mathcal{S}$ randomly generates $x, \beta \in_R \mathbb{Z}_p$ and sets $g' = g^\beta, X = e(g, g)^x$. $\mathcal{S}$ also moderates the hash functions H, H_I, H_t as random oracles. By controlling the random oracles, $\mathcal{S}$ can easily simulate the transcript $\mathcal{T}$.

In the Extraction Oracle, we use the same technique as in [4] to extract r from the user and set $r' = rr_{\mathbb{ID},1}$. $\mathcal{S}$ can generate $t_{\mathbb{ID}}$ easily, as it knows x, β.

$\mathcal{S}$ can also simulate the transcript $x_{\mathbb{ID}}$ by controlling the random oracle. □

Theorem 3. *The advantage of an adversary in the FindKey game is negligible for the proposed scheme if the discrete log problem in $\mathbb{G}$ is hard.*

Proof: The proof follows along the lines of the proof for the FindKey game in [4], except that we are in the adaptive chosen identity attack model.

Let $\mathcal{A}$ denote a PPT algorithm that has a non-negligible advantage ϵ in winning the FindKey game, we construct another PPT algorithm $\mathcal{B}$ that breaks the discrete log assumption in $\mathbb{G}$ with a non-negligible probability. $\mathcal{B}$ proceeds as follows.

$\mathcal{B}$ runs the algorithm $\mathcal{A}$ and gets the public parameters $mpk = (g, g_1, g', h_1, h_2, h_3, X, H, H_I, H_t, \mathcal{M}, \mathcal{T})$. $\mathcal{B}$ pass h_1 to its challenger and gets a challenge $R \in \mathbb{G}$. $\mathcal{B}$'s goal is to find $r \overset{def}{=} \log_{h_1} R$.

Assume adversary $\mathcal{A}$ makes at most q_I extract queries, after getting a challenge R from the challenger, $\mathcal{B}$ selects $i \overset{R}{\leftarrow} \{1, 2, 3, ..., q_I\}$. For each $1 \leq j \leq q_I$, if $j = i$, $\mathcal{B}$ sets $R_j = R$, otherwise, $\mathcal{B}$ randomly selects $r_j \in \mathbb{Z}_p$ and sets $R_j = h_1^{r_j}$.

$\mathcal{B}$ answers $\mathcal{A}$'s queries as follows:

- Extract queries. When $\mathcal{A}$ performs an extract query on an identity ID_j, $\mathcal{B}$ rejects the query if ID_j has already been created. Otherwise, $\mathcal{B}$ performs the extract protocol as follows: if $j \neq i$, $\mathcal{B}$ runs the extract protocol as usual; if $j = i$, $\mathcal{B}$ gives R to $\mathcal{A}$ together with a zero-knowledge proof. The zero-knowledge proof is generated by rewinding $\mathcal{A}$. Note that here we require $\mathcal{A}$ to commit to the challenge before the zero-knowledge proof is carried out so that when rewinding $\mathcal{A}$ the same challenge will be used by $\mathcal{A}$. In this way, $\mathcal{B}$ is able to simulate a proof by running the simulator of the zero knowledge proof system without the knowledge of r. After receiving $(r', h'_{\mathbb{ID}_i}, t'_{\mathbb{ID}_i}, r_{\mathbb{ID}_i,2}, h_{\mathbb{ID}_i,2}, r_{\mathbb{ID}_i,3}, h_{\mathbb{ID}_i,3})$ and $\pi_{\mathbb{ID}_i}$ from $\mathcal{A}$, $\mathcal{B}$ verifies $\pi_{\mathbb{ID}_i}$ and runs a key sanity check by testing if $e(h_{\mathbb{ID}_i,t}, g_1/g^{\mathbb{ID}_i}) = e(h_t g^{-r_{\mathbb{ID}_i,t}}, g)$ for $t = 2, 3$. For $t = 1$, $\mathcal{B}$ tests if $e(h'_{\mathbb{ID}_i}, g_1/g^{\mathbb{ID}_i}) = e(R_i g^{-r'}, g)$. If any of these tests fails, $\mathcal{B}$ aborts with failure, otherwise, ID_i is added to L_1, notice that $\mathcal{B}$ cannot derive the final user secret key usk_{ID_i} in this case.

- Corruption queries. When $\mathcal{A}$ performs a corruption query on identity $\hat{\mathsf{ID}}$, if $\hat{\mathsf{ID}} \in L_1$:
  (a) $\hat{\mathsf{ID}} = \mathsf{ID}_i$, $\mathcal{B}$ aborts with failure
  (b) $\hat{\mathsf{ID}} \neq \mathsf{ID}_i$, $\mathcal{B}$ returns $usk_{\hat{\mathsf{ID}}}$ to $\mathcal{A}$
  otherwise, $\mathcal{B}$ rejects the query.

Finally, if $\mathcal{B}$ does not abort the game, with probability ϵ, $\mathcal{A}$ will output a decryption key (passing the key sanity check) usk'_{ID_n} which has the same family number with usk_{ID_n} in L_1. If $n = i$, then $\mathcal{B}$ can calculate $r = r'/f'_{n_{\mathbb{ID}_i}}$ where $f'_{n_{\mathbb{ID}_i}}$ is the key family number of usk'_{ID_i}. It is obvious that $n = i$ implies that usk_{ID_i} is in L_1 and $\mathcal{B}$ does not abort in the game. Since i is randomly chosen, $\mathcal{B}$'s success probability in solving the discrete log problem in $\mathbb{G}$ is at least $\frac{\epsilon}{q_I}$. □

Theorem 4. *The advantage of an adversary in the* SanityCheck *game is negligible for the proposed scheme.*

Proof: *(sketch)* Let the output of $\mathcal{A}$ be mpk, ID^*, $C = (C_1, C_2, C_3, C_4, C_5)$, $usk_{\mathsf{ID}^*} = \{r, r_{\mathbb{ID},1}, r_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},1}, h_{\mathbb{ID},2}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}}\}$ and $\widetilde{usk}_{\mathsf{ID}^*} = \{\widetilde{r}, \widetilde{r}_{\mathbb{ID},1}, \widetilde{r}_{\mathbb{ID},2}, \widetilde{r}_{\mathbb{ID},3}, \widetilde{h}_{\mathbb{ID},1}, \widetilde{h}_{\mathbb{ID},2}, \widetilde{h}_{\mathbb{ID},3}, \widetilde{t}_{\mathbb{ID}}, \widetilde{\pi}_{\mathbb{ID}}\}$. $\mathcal{A}$ wins implies that condition (1) - (5) defined in section 4.2 are all fulfilled.

Condition (1) implies

$$\begin{aligned} e(h_{\mathbb{ID},i}, g_1/g^{\mathbb{ID}}) &= e(h_i g^{-r_{\mathbb{ID},i}}, g) \\ \Rightarrow\ e(h_{\mathbb{ID},i}, g^{\alpha-\mathbb{ID}}) &= e(h_i g^{-r_{\mathbb{ID},i}}, g) \\ \Rightarrow\ e((h_{\mathbb{ID},i})^{\alpha-\mathbb{ID}}, g) &= e(h_i g^{-r_{\mathbb{ID},i}}, g) \\ \Rightarrow\ (h_{\mathbb{ID},i})^{\alpha-\mathbb{ID}} &= (h_i g^{-r_{\mathbb{ID},i}}) \\ \Rightarrow\ h_{\mathbb{ID},i} &= (h_i g^{-r_{\mathbb{ID},i}})^{\frac{1}{\alpha-\mathbb{ID}}} \qquad \text{for } i = 1,2,3 \end{aligned} \tag{4}$$

Similarly, condition (2) implies

$$\widetilde{h}_{\mathbb{ID},i} = (h_i g^{-\widetilde{r}_{\mathbb{ID},i}})^{\frac{1}{\alpha-\mathbb{ID}}} \qquad \text{for } i = 1,2,3 \tag{5}$$

Condition (3) and (4) imply that C_5 verifies (that is, C_5 is a valid SPK). That is, in the random oracle model, the simulator can extract s such that

$$C_1 = (g^{\alpha-\mathbb{ID}})^s \qquad \text{and} \qquad C_2 = e(g,g)^s \tag{6}$$

Condition (5) implies that

$$C_3 \cdot e(C_1, h_{\mathbb{ID},1}) \cdot {C_2}^{r_{\mathbb{ID},1}} \neq C_3 \cdot e(C_1, \widetilde{h}_{\mathbb{ID},1}) \cdot {C_2}^{\widetilde{r}_{\mathbb{ID},1}} \tag{7}$$

We have

$$\begin{aligned} LHS &= C_3 \cdot e(C_1, h_{\mathbb{ID},1}) \cdot {C_2}^{r_{\mathbb{ID},1}} \\ &= C_3 \cdot e\Big((g^{\alpha-\mathbb{ID}})^s, (h_1 g^{-r_{\mathbb{ID},1}})^{\frac{1}{\alpha-\mathbb{ID}}}\Big) \cdot e(g,g)^{s\cdot r_{\mathbb{ID},1}} \text{ from equation (4) and (6)} \\ &= C_3 \cdot e(g^s, h_1 g^{-r_{\mathbb{ID},1}}) \cdot e(g,g)^{s\cdot r_{\mathbb{ID},1}} \\ &= C_3 \cdot e(g, h_1)^s \cdot e(g,g)^{s\cdot(-r_{\mathbb{ID},1})} \cdot e(g,g)^{s\cdot r_{\mathbb{ID},1}} \\ &= C_3 \cdot e(g, h_1)^s \end{aligned}$$

Similarly we have

$$\begin{aligned} RHS &= C_3 \cdot e(C_1, \widetilde{h}_{\mathbb{ID},1}) \cdot C_2^{\widetilde{r}_{\mathbb{ID},1}} \\ &= C_3 \cdot e(g, h_1)^s = LHS \end{aligned} \tag{8}$$

However, equation (8) contradicts to equation (7). Thus $\mathcal{A}$ wins the game only with negligible probability. □

Theorem 5. *The advantage of an adversary in the* Retrievability *game is negligible for the proposed scheme.*

Proof: *(sketch)* Let the output of $\mathcal{A}$ be mpk, ID^*, $C = (C_1, C_2, C_3, C_4, C_5)$, $usk_{\mathsf{ID}^*} = \{r, r_{\mathbb{ID},1}, r_{\mathbb{ID},2}, r_{\mathbb{ID},3}, h_{\mathbb{ID},1}, h_{\mathbb{ID},2}, h_{\mathbb{ID},3}, t_{\mathbb{ID}}, \pi_{\mathbb{ID}}\}$ and $\widetilde{usk}_{\mathsf{ID}^*} = \{\widetilde{r}, \widetilde{r}_{\mathbb{ID},1}, \widetilde{r}_{\mathbb{ID},2}, \widetilde{r}_{\mathbb{ID},3}, \widetilde{h}_{\mathbb{ID},1}, \widetilde{h}_{\mathbb{ID},2}, \widetilde{h}_{\mathbb{ID},3}, \widetilde{t}_{\mathbb{ID}}, \widetilde{\pi}_{\mathbb{ID}}\}$ such that $r_{\mathbb{ID},1} \neq \widetilde{r}_{\mathbb{ID},1}$. $\mathcal{A}$ wins implies that condition (1) - (4) defined in section 4.2 are all fulfilled.

Condition (1) and (2) implies that the PK on $\pi_{\mathbb{ID}}$ and $\widetilde{\pi}_{\mathbb{ID}}$ are sound. That is, $\left(\frac{t_{\mathbb{ID}}^{\tilde{K}}}{\tilde{t}_{\mathbb{ID}}^{K}}\right)^{\frac{1}{\tilde{K}-K}} = g^x$. Condition (3) implies that $r_{\mathbb{ID},1} \neq \widetilde{r}_{\mathbb{ID}_1}$, that is, $K := H_t(g^r, r_{\mathbb{ID},1} r) \neq \widetilde{K} := H_t(g^{\widetilde{r}}, \widetilde{r}_{\mathbb{ID},1}\widetilde{r})$. Condition (4) implies that either

1. $K \neq \tilde{K}$; or
2. $X \neq e(g, g^x)$ where g^x is computed from equation (1); or
3. $g_1 \neq g^\alpha$ where α is computed from equation (2) and (3)

Case (1) happens with negligible probability, due to the collision resistance property of the hash function H_t. Case (2) happens with negligible probability, due to the soundness of SanityCheck which has been proven above. Case (3) also happens with negligible probability, due to the security of the verifiable encryption scheme [9].

Combining all cases, the adversary only has negligible advantage to win the game. □

6 Conclusion

In this paper, we firstly identified a security issue of Goyal's work in [4, 5]. We then proposed a way to fix it. Then, we took one step further than Goyal's work by proposing a Traceable and Retrievable IBE. In our notion, the PKG's master secret key is retrieved automatically if more than one user secret key are released. We presented a formal model to capture this idea, and proposed a concrete scheme based on Gentry's IBE [3]. We believe that the model we proposed in this paper may be more appealing in practice as our model encourages the PKG to strengthen their security system. If someone can successfully release an additional user secret key, it means that his security is also compromised.

Acknowledgements

We would like to thank the anonymous reviewers of ACNS 2008 for their suggestions and invaluable comments to improve this paper.

References

1. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
2. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
3. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
4. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–448. Springer, Heidelberg (2007)
5. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. Cryptology ePrint Archive, Report 2007/368 (2007); revised and extended version of [4], http://eprint.iacr.org/2007/368
6. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 235–251. Springer, Heidelberg (1990)
7. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report 260, Institute for Theoretical Computer Science, ETH Zurich (1997)
8. Cramer, R., Damgård, I., MacKenzie, P.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–373. Springer, Heidelberg (2000)
9. Camenisch, J., Damgård, I.: Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000)