Resistance to Pirates 2.0: A Method from Leakage Resilient Cryptography

Duong Hieu Phan[1,2] and Viet Cuong Trinh[1]

[1]LAGA, University of Paris 8
[2]ENS / CNRS / INRIA

Abstract. In the classical model of traitor tracing, one assumes that a traitor contributes its entire secret key to build a pirate decoder. However, new practical scenarios of pirate has been considered, namely Pirate Evolution Attacks at Crypto 2007 and Pirates 2.0 at Eurocrypt 2009, in which pirate decoders could be built from sub-keys of users. The key notion in Pirates 2.0 is the anonymity level of traitors: they can rest assured to remain anonymous when each of them only contributes a very small fraction of its secret information. This scenario encourages dishonest users to participate in collusion and the size of collusion could become very large, possibly beyond the considered threshold in the classical model. There are numerous attempts to deal with Pirates 2.0 each of which only considers a particular form of Pirates 2.0. In this paper, we propose a method for fighting Pirates 2.0 in any form.
Our method is based on the researches in key-leakage resilience. It thus gives an interesting and rather surprised connection between the rich domain of key-leakage resilient cryptography and Pirates 2.0. We first formalize the notion of key-leakage resilient revoke system and then identify sufficient conditions so that a key-leakage resilient revoke scheme can resist Pirates 2.0 in any form. We finally propose a construction of a secure key-leakage resilient identity-based revoke system that fulfills the required conditions. The main ingredient in the construction relies on the identity-based encryption with wildcards (WIBE) and our construction of key-leakage resilient WIBE could be useful in its own right.

Keywords: Pirates 2.0, Leakage-resilience, wildcards, revocation.

1 Introduction

In a system of secure distribution of digital content, a center broadcasts encrypted content to legitimate recipients. *Broadcast encryption systems*, independently introduced by Berkovits [5] and Fiat-Naor [16], enable a center to encrypt a message for any subset of legitimate users while preventing any set of revoked users from recovering the broadcasted information. Moreover, even if all revoked users collude, they are unable to obtain any information about the content sent by the center. *Traitor tracing schemes*, introduced in [10], enable the center to trace users who collude to produce pirate decoders. *Trace and Revoke systems* [24,23] provide the functionalities of both broadcast encryption and traitor tracing.

In the classical model of tracing traitors, one assumes that a traitor contributes its entire secret key to build a pirate decoder. However, new practical scenarios of pirate has been considered, namely Pirate Evolution Attacks [19] and Pirates 2.0 [6], in which pirate decoders could be built from sub-keys of users. The notion of anonymity has been put forth in Pirates 2.0 and it is shown that if each user only contributes a very small fraction of its secret information, he can rest assured to remain anonymous. This scenario encourages dishonest users to participate in collusion and the size of collusion could becomes very large, beyond the considered threshold in the classical model.

There are some methods aiming to fight against pirates 2.0 attacks [11,27,30] but none of these works considers a general form of leakage of secret keys. In fact, it is assumed in these methods that the dishonest users leak the entire information of some sub-keys which could be used in the encryption procedure. It was also mentioned in these papers that a method for dealing with Pirates 2.0 in any form of leakage rather than contributing the whole information about some sub-keys is a open and challenging problem. We solve this problem by considering any strategy of contributing information. The key point in our analysis is to quantify the leaked information (via the conditional entropy) of the secret keys of the users before and after each round of contribution.

In order to fight against the public contribution of traitors, we study a method that forces traitors to contribute a large amount of their secret information (and hence the traitors can no longer remain anonymous) by proving that if the traitors only contribute small parts of their keys, the built pirate decoder cannot be useful. This leads us to the consideration of key-leakage resilient in revoke schemes.

Leakage resilient cryptography has been a very rich domain of research in the recent years, a non-exhaustive list of works can be found in [17,22,12,9,14,26,15,28,25,18,4,8,20]. Under this framework, the adversary is allowed to specify an efficiently computable leakage function and learn the output of the function applied to the secret key and possibly other internal state information at specified moments in the security game. Our idea is to reduce a Pirates 2.0 to an adversary that breaks the security of a key-leakage resilient revoke scheme in which the contributive function in Pirates 2.0 is used as the computable leakage function and the high anonymity level in Pirates 2.0 is linked to the level of leaked information.

1.1 Contribution

Theoretical result. We formalize the key-leakage resilient security model for a revoke system, which enhances its classical security model. We then prove that any key-leakage resilient revoke system satisfying the following conditions will resist Pirates 2.0 in any form:

– any user's secret key is a high independent source, *i.e.,* it has a high entropy even under the condition that all the keys of the others users are known.
– resilience to a sufficient high level of leakage at secret keys of users.

Intuitively, the first condition assures that the secret keys of users are sufficiently independent each from the others and the second condition implies that the users should contribute a high information about its key to produce an useful decoder. Combining the two conditions, the users have to contribute high information of their own independent sources and thus lose their anonymity.

Construction. In order to apply the above result, we present a secure key-leakage resilient identity-based revoke scheme that fulfills the required conditions to resist Pirates 2.0. Because our construction is based on the identity-based encryption with wildcards (WIBE) [2,1] in the similar way to [27], it turns out that the main obstacle is to construct a key-leakage resilient WIBE which could be useful in its own right. This is not a trivial work and is achieved in successive steps:

– The security model of a key-leakage resilient WIBE generalizes the full security of a WIBE by allowing the adversary to make additional leak queries. Our first step is then to construct an efficient fully secure WIBE. Fortunately, with the recent dual system encryption technique in [29], it's relativelly simple to construct a variant of the Boneh-Boyen-Goh's WIBE ($\mathsf{BBG-WIBE}$) [2] scheme that is fully secure with a very efficient reduction that avoids a loss of an exponential factor in hierarchical depth as in the classical method of reducing the full security of WIBE to the full security of the underlying HIBE in [2].
– Inspired by the security proof technique of the key-leakage resilient HIBE in [20], our second and main step is to transform this variant of fully secure $\mathsf{BBG-WIBE}$ to a secure key-leakage resilient WIBE. Some carefulness should be taken into account in the security proof because WIBE is a generalization of HIBE and the adversary has more freedom in attacking WIBE than in attacking HIBE. Our construction of the first key-leakage resilient WIBE could have its own impact.

1.2 Related works

Identity-based traitor tracing scheme was proposed by Abdalla et al [3] in which one can distribute content to various groups of users by taking as input the identity of the targeted group. *Identity-based trace and revoke schemes* (IDTR) in [27] extended this model to allow the center to be capable of revoking any subgroup of users.

Identity-based encryption with wildcards (or WIBE for short) was proposed by Abdalla *et al* [2] and can be seen as a generalization of HIBE. This primitive is related to broadcast encryption in the sense that the encryption is targeted to a group of users rather than to only one user. However, the targeted set of users in WIBE follows a pre-determined structure while a broadcast encryption should be able to target arbitrary group of users. Naturally, WIBE could then be used as a sub-structure to construct trace and revoke systems. This approach has been used in different ways, namely under the code-based framework [3,30], and under the tree-based framework [27]. Our construction is under the tree-based framework as in [27] but with a key-leakage resilient WIBE.

2 Sufficient Condition for Fighting Pirates 2.0

In this section, we identify the sufficient condition for a key-leakage resilient revoke system to resist Pirates 2.0 attack. We first introduce the formalization of a key-leakage resilient revoke system, the review the Pirates 2.0 in the information theory and finally establish a sufficient condition on the independent entropy of the secret keys in a key-leakage resilient revoke system to exclude the threat of Pirates 2.0 in any form.

2.1 Key-Leakage Resilient Revoke System

We recall the definition of a revoke scheme. Formally, a revoke scheme consists of four polynomial-time algorithms (**Setup**, **KeyDer**, **Enc**, **Dec**):

Setup$(1^k, N_u)$**:** Takes as inputs the security parameter 1^k and the number of users N_u. This algorithm generates a master public key mpk and a master secret key msk.

KeyDer(msk, i)**:** Takes as inputs an indices i of user and the master secret key msk, the key extraction algorithm generates a user secret key sk_i.

Enc$(\mathsf{mpk}, \mathcal{R}, M)$**:** The encryption algorithm which on inputs of the master public key mpk, a revocation list $\mathcal{R}$ of revoked users in the system, and a message M outputs a ciphertext C.

Dec(sk_i, C)**:** The decryption algorithm which on input of a user secret key sk_i and a ciphertext C outputs a plaintext message M, or $\perp$ to indicate a decryption error.
For correctness we require that $\mathbf{Dec}(sk_i, \mathbf{Enc}(\mathsf{mpk}, \mathcal{R}, M)) = M$ with probability one for all $i \in \mathbb{N} \setminus \mathcal{R}$, $M \in \{0,1\}^*$, $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathbf{Setup}(1^k, N_u)$ and $sk_i \xleftarrow{\$} \mathbf{KeyDer}(\mathsf{msk}, i)$.

We now present the security model for a (ℓ_{SK})-key-leakage resilient revoke scheme (each user leaks maximum ℓ_{SK} bits on his secret key SK).

Setup: The challenger takes a parameter k, a maximum number of users N_u and runs $setup(1^k, N_u)$ algorithm. The master public key mpk is passed to the adversary. Also, it sets the set of revoked users $\mathcal{R} = \emptyset$, $\mathcal{T} = \emptyset$, note that $\mathcal{R} \subseteq \mathcal{I}$, and $\mathcal{T} \subseteq \{\mathcal{I} \times \mathcal{SK} \times \mathcal{N}\}$ (users indices - secret key of users - leaked bits). Thus initially, no leakage on each secret key.

Phase 1: The adversary can be interleaved in any possible way to request three types of query:

1. **Create**(i): The challenger initially scans $\mathcal{T}$ to find the indices i. If this indices exists in $\mathcal{T}$, it responds with $\perp$.
   Otherwise, the challenger makes a call to $\mathbf{KeyDer}(\mathsf{msk}, i) \to sk_i$ and adds the tuple (i, sk_i, 0) to the set $\mathcal{T}$.
2. **Leak**$(i,\ f)$ In this query, the adversary requests leakage from a key that has indices i with a polynomial-time computable function f of constant output size. The challenger scans $\mathcal{T}$ to find the specified indices. It is of the form (i, sk_i, L). It checks if $L + \mid f(sk_i) \mid \leq \ell_{SK}$. If this is true, it responds with $f(sk_i)$ and updates the L in the tuple with $L + \mid f(sk_i) \mid$. If the checks fails, it returns $\perp$ to the adversary.
3. **Reveal**(i): Now the adversary requests the entire key with indices i. The challenger scans $\mathcal{T}$ to find the requested entry. Let's say the tuple is (i, sk_i, L). The challenger responds with sk_i and adds the indices i to the set $\mathcal{R}$.

Challenge: The adversary submits two equal length messages M_0, M_1. The challenger picks a random bit $b \in \{0,1\}$ and set $C = \text{Encrypt}(\mathsf{msk}, \mathcal{R}, M_b)$. The ciphertext C is passed to the adversary.

Phase 2: This is identical to phase 1 except that the adversary is not allowed to ask **Reveal**(i) query in which $i \notin \mathcal{R}$.
Guess: The adversary outputs a guess b' and wins the game if $b' = b$.

Definition 1. A revoke scheme is (ℓ_{SK})-key-leakage resilient secure if all probabilistic polynomial-time adversaries (called PPT adversaries for short) have at most a negligible advantage in winning the above security game.

2.2 Pirates 2.0

In the model of pirates 2.0 attacks [6], traitors collaborating in a public way and use the same strategy to display *part of their secret keys* in a public place at their discretion; pirate decoders are then built from this public information. The distinguishing property of pirates 2.0 attacks is that traitors only contribute partial information about their secret key material which suffices to produce (possibly imperfect) pirate decoders while allowing them to remain anonymous. Both pirates and traitors can keep track of all of the information that was contributed to the public.

The basic idea behind Pirates 2.0 attacks is that traitors are free to contribute some piece of secret data as long as several users of the system could have contributed exactly the same information *following the same (public) strategy*: this way, they are able to remain somewhat anonymous. The leakage information is formalized via extraction function which is any efficiently computable function f on the space of the secret keys and a traitor u is said to be *masked* by a user u' for an extraction function f if $f(sk_u) = f(sk_{u'})$. The anonymity level is meant to measure exactly how anonymous they remain. This is defined in [6] as follows.

Definition 2 (Anonymity Level). The level of anonymity of a traitor u after a contribution $\cup_{1 \le i \le t} f_i(sk_u)$ is defined as the number α of users masking u' for each of the t extraction functions f_i simultaneously:

$$\alpha = \#\{u' \mid \forall i,\ f_i(sk_u) = f_i(sk_{u'})\} \ .$$

Useful pirate decoder An pirate decoder is useful if it can decrypt a very large set of ciphertexts for almost all the target sets chosen by the broadcastor. We only need a minimum condition of usefulness on the pirate decoder: the pirate has to be able to output a target set S so that the pirate decoder can decrypt ciphertexts for this target set S with a non-negligible probability (the probability is taken on the randomness used for generating a ciphertext for S). In fact, if it is hard to expose such a set S then the pirate decoder can only decrypt with a negligible probability the ciphertexts outputted by the broadcaster and cannot be useful. Our objective is to construct a scheme that is immune even to these pirates of minimum usefulness.

Definition 3 (Pirates 2.0). A traitor tracing scheme is said to be vulnerable against a Pirates 2.0 attack if:

– there is a construction of a pirate decoder from information published by traitors in such a way that the traitor rest assured to have an anonymity level of $\alpha > 1$.
– the pirate is able to specify at least one target set S so that the produced pirate decoder can decrypt ciphertexts for this target set S with a non-negligible probability.

2.3 Pirates 2.0 viewed from the information theory

We aim to re-explain the way Pirates 2.0 works in [6] under the information theory. This is also the basic starting point so that we can establish a sufficient condition for a scheme to resist Pirates 2.0 in the next sub-section. In a revoke scheme, when a user joins the system, its key is generated and has some entropy. However, as keys of users could be correlated, the user can contribute some correlated information without the risk being identified. The user really lose its anonymity when he contributes its independent secret information that the other users don't have. More formally, these are entropy conditioned on the information about the other users' keys. Let us first recall some classical definitions about entropy.

Definition 4. Let X be a random variable. The min-entropy of X is

$$H_\infty(X) = \min_x - \log(\Pr[X = x]) = - \log(\max_x \Pr[X = x])$$

We say that X is a k-source if $H_\infty(X) \geq k$.

The high min-entropy is used rather than the Shannon entropy in cryptography for describing good distributions for the keys. In fact, the conventional notion in cryptography is the intuitive notion of "guessability" and a distribution X has min-entropy of k bits if even an unbounded adversary cannot guess a sample from X with probability greater than 2^{-k}.

However, in context of Pirates 2.0, a high min-entropy is not enough because the keys could be correlated. We should thus need to define how many information of the key a user has that is independent to the keys of the others users. This is quantified via the conditional min-entropy.

Definition 5. Let X, E be a joint distribution. Then we define the min-entropy of X conditioned on E-denoted $H_\infty(X|E)$ as

$$H_\infty(X|E) = - \log \max_e [\max_x \Pr[(X|E = e)]]$$

We say that X is a k-independent source of E if $H_\infty(X|E) \geq k$.

For the purpose of randomness extraction, Dodis et. al. [13] observed that because E is not under adversarial control, it suffices to consider an average min-entropy as $H_\infty(X|E) = \log \mathrm{E}[\max_x \Pr[(X|E = e)]]$. In our setting, the users can choose some strategies to contribute their information, the distribution E is not totally independent from the adversarial control, we need therefore to consider the conditional min-entropy rather than the average min-entropy. Fortunately, we will see later in our construction that the secret keys of users are sufficiently independent each from the others, the use of the conditional min-entropy is appropriate. We define the independence between the secret keys in a revoke system as follows.

Definition 6 (Independent Source). In an revoke system of N_u users, let X_i be the distribution outputted by the key generation for the user i and let $E = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{N_u}, \mathsf{pub})$ where pub denotes the distribution of the public parameters in the system. Then we say that the key of user i is a k-independent source if $H_\infty(X_i|E) \geq k$.

The key of user i is a k-independent source if it has k-bit entropy independently from the keys of the others users and from all the public information of the systems.

We now review the Pirates 2.0 in the context of Complete Subtree resumed in Figure 2.3. For a D-level tree, each user's key is a $(D \times \lambda)$-source but only a λ-independent source because each user only has an independent sub-key at the leaf. Therefore, even if an user contributes $((D-1) \times \lambda)$ entropy of its key, the remained information could still be a λ-independent source. Without leaking any independent entropy, the user could remain anonymous at a level $\alpha > 1$ (because at least two different users can have the same contributive information). In the example in Figure 2.3, the user U is assigned 5 sub-keys corresponding to the nodes from the root to the leaf. The user U can contribute a key S_4 and specifies the target set at S_4 that covers 4 users of the sub-tree rooted at S_4. An pirate decoder with only one key at S_4 can decrypt the ciphertext for the chosen target set S_4 with non-negligible probability while preserving an anonymity level $\alpha = 4$ for the contributor and therefore, the scheme is vulnerable against the Pirates 2.0. [1]

[1] We note that an useful Pirates 2.0 in practice should do much more than this Pirates 2.0 with minimum usefulness because it should deal with a large type of target set and different strategies of the broadcastor. However, as our objective is to construct a scheme that is immune to any Pirates 2.0, we consider here the minimum usefulness of Pirates 2.0.

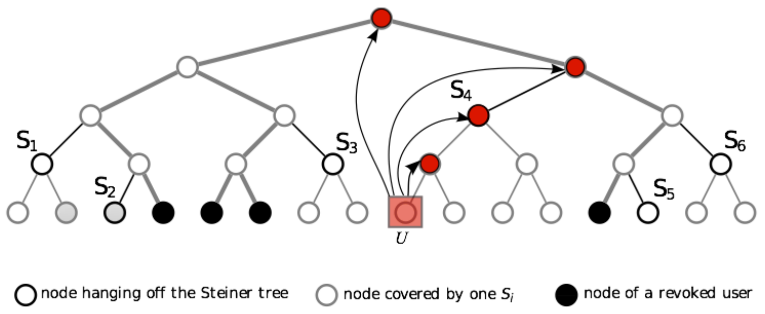

Fig. 1. An example of a complete subtree scheme where the center covers all non-revoked users with the nodes $S_1, \ldots, S_6$. A user is a leaf on the binary tree where each node is assigned to a long-lived randomly chosen key. Each user possesses all the long-lived keys of the nodes on the path from the user's leaf to the root.

2.4 Key-Leakage Resilience vs. Pirates 2.0

We are now ready to prove a sufficient condition so that a key-leakage resilient revoke scheme is immune to Pirates 2.0 attacks. This is the main result of this section. In the next section, we will construct a key-leakage resilient revoke scheme that fulfills the required sufficient condition.

Theorem 7. *Let Π be a (ℓ_{SK})-key-leakage resilient revoke system of N_u users in which each user's key has length of m bit and is a m'-independent source. If $\alpha = \frac{N_u}{2^{\ell_{SK}+m'-m}} \leq 1$, then Π is immune to any Pirates 2.0 attack.*

Proof.

Proposition 8. *In a Pirates 2.0 attack, if an user leaks k bits of his secret key to the public domain then his anonymity level is at most $\frac{N_u}{2^{k+m'-m}}$.*

Proof. Intuitively, as the key of the user u is a high independent source even when the others users contribute their whole secret keys, if u leaks too much information on its key then it will also leak many independent information and loses its anonymity.

Formally, following the definition 2 of anonymity level in pirates 2.0, assume that a user u contributes k bits information L_u of his secret key sk_u to the public domain, we need to compute the probability for an user u' to contribute exactly the same information as the user u, at each periode of time i.

– At time 0: u contribute nothing to the public domain. Let $E_i = (\cup_{w \neq u} sk_w, \mathsf{pub}_i)$ where pub_i denotes the public information at the time i which contains the publics parameters of the system plus contributed information of the users after the time $i-1$. Because each user's key is a m'-independent source: $H_\infty(sk_u|E_0) \geq m'$.
– At time i: u contributes his secret informations $L_u^i = f_i(sk_u, \mathsf{pub}_i)$ to the public domain by leaking k_i bits of his secret keys. If we denote k_i^{in} the number of independent bits that the user u losses in time i, *i.e.,* $k_i^{in} = H_\infty(sk_u|E_i) - H_\infty(sk_u|E_{i-1})$, then the probability that u' could contribute exactly the same information L_u^i is at most $\frac{1}{2^{k_i^{in}}}$. Note that E_0 and thus E_i already contain $\cup_{w \neq u} sk_w$, *i.e.,* all the contributed information of the other users are already contained in E_i (for all i), the k_i^{in} independent bits are among k_i bit that the user u leaks at the time i.

At the end, after the time t, the user u contributes to the public domain by totally leaking $k = k_1 + \cdots + k_t$ bits of its secret information. By the above computation, the probability that an user u' can contribute exactly the same total information like u is at most $\prod_{j=1}^{t} \frac{1}{2^{k_j^{in}}}$, and

$$\sum_{j=1}^{t} k_j^{in} = H_\infty(sk_u|E_0) - H_\infty(sk_u|E_t)$$

Because the bit length of the secret key sk_u is m and the user u leaks k bits, we deduce that $H_\infty(sk_u|E_t) \leq m - k$ and therefore $\sum_{j=1}^{t} k_j^{in} \geq m' - (m - k) = k + m' - m$ which implies that the probability that an user u' can contribute exactly the same information like u as required in Pirates 2.0 is at most $\frac{1}{2^{k+m'-m}}$ and the anonymity level of u cannot be assured to be higher than $\frac{N_u}{2^{k+m'-m}}$. □

Proposition 9. *Let Π be a (ℓ_{SK})-key-leakage resilient revoke scheme. If each user leaks no more than ℓ_{SK} bits of his secret key to the public domain, then one can not produce a Pirates 2.0 decoder.*

Proof. We suppose by contradiction that there is an Pirates 2.0 $\mathcal{A}$ against Π in which each user leaks no more than ℓ_{SK} bits of his secret key to the public domain, then we build an algorithm $\mathcal{B}$ that breaks the security of Π in the context of key leakage resilience.

Algorithm $\mathcal{B}$ simulates $\mathcal{A}$ and makes use of the outputs of $\mathcal{A}$ to break the security of Π. It works as follows:

– At time 0: users contribute nothing to the public domain.
– At time 1: suppose that an user u decides to contribute $L_u^1 = f_1(sk_u)$ bits to the public domain by using a strategy f_1 where f_1 is a polynomial-time computable function, $\mathcal{B}$ requests the leak query $(u, g_1 := f_1)$ to his challenger and forwards the result to $\mathcal{A}$.
– At any time i: suppose that an user u decides to contribute $L_u^i = f_i(sk_u, I)$ bits to the public domain, where I is the public collected information after the time $i-1$. At this stage, $\mathcal{B}$ defines a polynomial-time computable function $g_{i,I}(sk_u) := f_i(sk_u, I)$, then requests the leak query $(u, g_{i,I})$ to his challenger and forwards the result to $\mathcal{A}$.
– When $\mathcal{A}$ outputs a pirate decoder and a target S so that the pirate decoder can decrypt ciphertexts for S with a non-negligible probability, $\mathcal{B}$ simply outputs $S^* = S$ and two different messages M_0, M_1 to his challenger. By using this pirate decoder, $\mathcal{B}$ can decrypt the challenge ciphertext with a non-negligible probability and thus break the security of the scheme.

We note that, since each user contributes maximum ℓ_{SK} bits to the public domain, $\mathcal{B}$ only need to ask in total at most ℓ_{SK} bits to his challenger. By definition, Π is then not ℓ_{SK}-key leakage resilient. □

The theorem immediately follows from the above two propositions. □

3 Key-Leakage Resilient Revoke Scheme Immune to Pirates 2.0

This section is devoted to construct a key-leakage resilient revoke scheme that fulfills the condition in Theorem 7 and thus is immune to Pirates 2.0 attacks. The construction is achieved via the following steps:

1. we first propose a variant of $\mathsf{BBG}-\mathsf{WIBE}$ scheme which is proven fully secure by using the dual system encryption technique.
2. we then construct a key-leakage resilient $\mathsf{BBG}-\mathsf{WIBE}$ scheme by employing the proof technique in [20] to the above $\mathsf{BBG}-\mathsf{WIBE}$. This is the most important step in the final construction.
3. we finally apply the generic transformation from a WIBE to an identity based trace and revoke scheme (denoted IDTR) in [27]. This results to a key-leakage resilient identity-based trace and revoke scheme (denoted KIDTR) that fulfills the condition in Theorem 7 and is immune to Pirates 2.0 attacks.

3.1 $\mathsf{BBG}-\mathsf{WIBE}$ in Composite Order Groups

In [21], Lewko and Waters apply the dual system encryption technique to prove the full security of the $\mathsf{BBG}-\mathsf{HIBE}$ scheme. This technique first splits the security game into $q+5$ games where q is the maximum number of queries that adversary makes. The first game is the real $\mathsf{BBG}-\mathsf{HIBE}$ security game and the final game gives no advantage for the adversary. Second, based on the three complexity assumptions $1, 2, 3$ in Appendix A, step by step they prove that these games are indistinguishable, this automatically avoids a loss of an exponential factor in hierarchical depth as in the classical method. This is achieved via the main concept of the nominal semi-functionality in the dual system encryption technique.

We follow their approach by applying the dual system encryption technique to construct a fully secure variant of the $\mathsf{BBG} - \mathsf{WIBE}$ scheme. The problem here is that the transformation from the $\mathsf{BBG} - \mathsf{HIBE}$ to the $\mathsf{BBG} - \mathsf{WIBE}$ needs to introduce additional components $(C_{3,i})$ in the ciphertext and these components demolish the nominal property because they are not *nominal* with respect to components (E_i) in semi-functional key. In order to retain the nominality, we should manage to impose the distribution of exponents of $\mathbb{G}_2$ part in $C_{3,i}$ and in E_i in a compatible way such that they are *nominal* with each other.

We provide the details about our construction of $\mathsf{BBG} - \mathsf{WIBE}$ scheme in composite order groups and the proof of its full security in Appendix B.

3.2 KWIBE: Key-Leakage Resilient WIBE

In the construction of key-leakage resilient HIBE in [20], the user's secret key is constructed from elements in subgroups $\mathbb{G}_1$ and $\mathbb{G}_3$. This leads to secret keys that are relatively low independent sources because they are only in subgroups $\mathbb{G}_1$ and $\mathbb{G}_3$. In order to enhance the independent sources of each user's secret key, in our construction of KWIBE, the secret keys are in the semi-functional form and each user's secret key is now a high independent source as a main part of the secret key is in the whole group $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_3$. Fortunately, this slightly change doesn't affect the functionality and the security of the scheme.

Construction from $\mathsf{BBG} - \mathsf{WIBE}$ The main point in proving the key-leakage resilience of HIBE in [20] is to show that the adversary cannot distinguish between two games $\mathsf{KeyLeak}_0$ and $\mathsf{KeyLeak}_1$ which are briefly described as follow. In the game $\mathsf{KeyLeak}_b$ game (for both $b = 0$ and $b = 1$), the adversary can choose to receive a normal key or a semi-functional key from each leak and reveal query for all keys except one key- called the challenge key. Concerning the challenge key, it is set to be a normal key in the game $\mathsf{KeyLeak}_0$ and a semi-functional key in the game $\mathsf{KeyLeak}_1$. We can realize that, in this technique of proving the security, there is no a significant difference between a HIBE attack and a WIBE attack. Indeed, the main difference between HIBE and WIBE is that an adversary against WIBE can ask more leak queries (for keys that *match* the challenge pattern) than an adversary against HIBE (who can only ask for keys which are *prefix* of the challenge identity). However, because the difference between two games $\mathsf{KeyLeak}_0$ and $\mathsf{KeyLeak}_1$ is only related to the challenge key which has the same form in both HIBE and WIBE, the proof in HIBE is well adapted to WIBE.

In order to make $\mathsf{BBG} - \mathsf{WIBE}$ resilient to key-leakage, in the following construction, we first impose the distribution of exponents of $\mathbb{G}_2$ part in $C_{3,i}$ and in E_i in a compatible way such that they are nominal with each other, then we choose compatibly some constants (as r_1, r_2, z_k, z_c) to keep the following properties:

– if $\overrightarrow{\Gamma}$ is orthogonal to $\overrightarrow{\delta}$ then the challenge key is well-distributed nominally semi-functional.
– if $\overrightarrow{\Gamma}$ is not orthogonal to $\overrightarrow{\delta}$, then the challenge key is truly semi-functional and well-distributed.

The construction is detailed as follows.

Setup$(1^\lambda) \rightarrow (\mathsf{mpk}, \mathsf{msk})$ The setup algorithm chooses a bilinear group $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_3$ of order $N = p_1 p_2 p_3$ (each subgroup $\mathbb{G}_i$ is of order p_i). We will assume that users are associated with vectors of identities whose components are elements of $\mathbb{Z}_N$. If the maximum depth of the WIBE is D, the setup algorithm chooses a generator $g_1 \xleftarrow{\$} \mathbb{G}_1$, a generator $g_2 \xleftarrow{\$} \mathbb{G}_2$, and a generator $g_3 \xleftarrow{\$} \mathbb{G}_3$. It picks $b, a_1, \ldots, a_D \xleftarrow{\$} \mathbb{Z}_N^{D+1}$ and sets $h = g_1^b, u_1 = g_1^{a_1}, \ldots, u_D = g_1^{a_D}$. It also picks

$n{+}1$ random exponents $\langle \alpha, x_1, x_2, \ldots, x_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$. The secret key is $\mathsf{msk} = (\alpha, a_1, \ldots, a_D)$, and the public parameters are:

$$\mathsf{mpk} = (N, g_1, g_3, h, u_1, \ldots, u_D, e(g_1, g_1)^{\alpha}, g_1^{x_1}, g_1^{x_2}, \ldots, g_1^{x_n})$$

KeyderSF$(\mathsf{msk}, (ID_1, ID_2, \ldots, ID_j), g_2, \mathsf{mpk})$ The key generation algorithm picks $n{+}1$ random exponents $\langle r, t_1, t_2, \ldots, t_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, $\overrightarrow{\rho} \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$ and $z_k, \rho_{n+3}, \ldots, \rho_{n+2+D-j} \overset{\$}{\leftarrow} \mathbb{Z}_N$, and $\overrightarrow{\gamma} = (\gamma_1, \ldots, \gamma_{n+2})$ in which $(\gamma_1, \ldots, \gamma_n, \gamma_{n+2}) \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, $\gamma_{n+1} = \gamma_{n+2}(z_k - \sum_{i=1}^{j} a_i ID_i)$. It outputs the secret key $SK = (\overrightarrow{K_1}, E_{j+1}, \ldots, E_D)$:

$$= \left(\left\langle g_1^{t_1}, g_1^{t_2}, \ldots, g_1^{t_n}, g_1^{\alpha} \left(h \cdot \prod_{i=1}^{j} u_i^{ID_i} \right)^{-r} \prod_{i=1}^{n} g_1^{-x_i t_i}, g_1^{r} \right\rangle * g_3^{\overrightarrow{\rho}} * g_2^{\overrightarrow{\gamma}}, \right.$$
$$\left. u_{j+1}^{r} g_3^{\rho_{n+3}} g_2^{\gamma_{n+2} a_{j+1}}, \ldots, u_D^{r} g_3^{\rho_{n+2+D-j}} g_2^{\gamma_{n+2} a_D} \right)$$

Note that, to run the **KeyderSF** algorithm one doesn't need to have g_2, he only need to have $X_2 \in \mathbb{G}_2$ or $X_2 X_3$ in which $X_2 \in \mathbb{G}_2, X_3 \in \mathbb{G}_3$.

Delegate $((ID_1, ID_2, \ldots, ID_j)$,SK',$ID_{j+1})$ Given a secret key SK' $= (\overrightarrow{K'}, E'_{j+1}, \ldots, E'_D)$ for identity $(ID_1, ID_2, \ldots, ID_j)$, this algorithm outputs a key for $(ID_1, ID_2, \ldots, ID_{j+1})$. It works as follow: It picks $n{+}1$ random exponents $\langle r', y_1, y_2, \ldots, y_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, $\overrightarrow{\rho}' \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$, and $\rho'_{n+3}, \ldots, \rho'_{n+1+D-j} \overset{\$}{\leftarrow} \mathbb{Z}_N$. It outputs the secret key $SK = (\overrightarrow{K_1}, E_{j+2}, \ldots, E_D)$:

$$= \left(\overrightarrow{K'_1} * \left\langle g_1^{y_1}, g_1^{y_2}, \ldots, g_1^{y_n}, h^{-r'} (E'_{j+1})^{-ID_{j+1}} \left(\prod_{i=1}^{j+1} u_i^{ID_i} \right)^{-r'} \prod_{i=1}^{n} g_1^{-x_i y_i}, g_1^{r'} \right\rangle * g_3^{\overrightarrow{\rho}'}, \right.$$
$$\left. E'_{j+2} u_{j+2}^{r'} g_3^{\rho'_{n+3}}, \ldots, E'_D u_D^{r'} g_3^{\rho'_{n+1+D-j}} \right)$$

Enc$(M, (P_1, P_2, \ldots, P_j))$ The encryption algorithm chooses $s \overset{\$}{\leftarrow} \mathbb{Z}_N$ and outputs the ciphertext:

$$CT = (C_0, \overrightarrow{C_1}, C_2)$$
$$= \left(M \cdot e(g_1, g_1)^{\alpha \cdot s}, \left\langle (g_1^{x_1})^s, \cdots, (g_1^{x_n})^s, g_1^s, (h \cdot \prod_{i \in \overline{W}(P)} u_i^{P_i})^s \right\rangle, (C_{2,i} = u_i^s)_{i \in W(P)} \right)$$

Dec(CT, SK) Any other receiver with identity $ID = (ID_1, ID_2, \ldots, ID_j)$ matching the pattern P to which the ciphertext was created can decrypt the ciphertext $CT = (C_0, \overrightarrow{C_1}, C_2)$ as follows First, he recovers the message by computing

$$\overrightarrow{C'_1} = \left\langle (g_1^{x_1})^s, \cdots, (g_1^{x_n})^s, g_1^s, (h \cdot \prod_{i \in \overline{W}(P)} u_i^{P_i})^s \cdot \prod_{i \in W(P)} (u_i^s)^{ID_i} \right\rangle$$

Finally, compute

$$e_{n+2}(\overrightarrow{K_1}, \overrightarrow{C'_1}) = e(g_1, g_1)^{\alpha s} \cdot e(g_1, u_1^{ID_1} \cdots u_j^{ID_j} h)^{-rs} \cdot e(g_1, u_1^{ID_1} \cdots u_j^{ID_j} h)^{rs} \cdot$$

$$\cdot \prod_{i=1}^{n} e(g_1, g_1)^{-x_i t_i s} \cdot \prod_{i=1}^{n} e(g_1, g_1)^{x_i t_i s} = e(g_1, g_1)^{\alpha s}$$

Notice that the $\mathbb{G}_2$ and $\mathbb{G}_3$ parts do not contribute because they are orthogonal to the ciphertext under e.

Security of Key-Leakage Resilient $\mathsf{BBG-WIBE}$ We introduce the definition of the security model of a ℓ_{SK}-key-leakage resilient WIBE, called $\mathsf{Leak-WIBE}$ security game, in Appendix C. In order to facilitate the presentation, let us first discuss about some notions that will be used in the proof of security:

Normal Key Functionality

Keyder$(\mathsf{msk}, (ID_1, ID_2, \ldots, ID_j), \mathsf{mpk})$. To create the normal key algorithm picks $n+1$ random exponents $\langle r, z_1, z_2, \ldots, z_n\rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, $\overrightarrow{\rho} \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$ and $\rho_{n+3}, \ldots, \rho_{n+2+D-j} \overset{\$}{\leftarrow} \mathbb{Z}_N$. It outputs the secret key $SK = (\overrightarrow{K_1}, E_{j+1}, \ldots, E_D)$:

$$= \Big(\Big\langle g_1^{z_1}, g_1^{z_2}, \ldots, g_1^{z_n}, g_1^{\alpha}\left(h \cdot \textstyle\prod_{i=1}^{j} u_i^{ID_i}\right)^{-r} \prod_{i=1}^{n} g_1^{-x_i z_i}, g_1^{r}\Big\rangle * g_3^{\overrightarrow{\rho}},$$
$$u_{j+1}^{r} g_3^{\rho_{n+3}}, \ldots, u_D^{r} g_3^{\rho_{n+2+D-j}}\Big)$$

Semi-Ciphertext Functionality

EncSF$(\mathrm{M}, \overrightarrow{P}) \to \widetilde{CT}$. This algorithm first calls the normal encryption algorithm **Enc**$(\mathrm{M}, \overrightarrow{P})$ to get the ciphertext $CT = (C_0, \overrightarrow{C_1}, C_2)$. Then it picks randomly $z_c \in \mathbb{Z}_N$, and $\overrightarrow{\delta} = (\delta_1, \ldots, \delta_{n+2})$ in which $(\delta_1, \ldots, \delta_{n+1}) \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, $\delta_{n+2} = \delta_{n+1}(z_c + \sum_{i \in \overline{W}(P)} a_i P_i)$, and outputs

$$\widetilde{CT} = \left(C_0, \overrightarrow{C_1} * g_2^{\overrightarrow{\delta}}, (C_{2,i} * g_2^{\delta_{n+1} \cdot a_i})_{i \in W(P)}\right)$$

The parameters in p_2 of $\widetilde{CT}$ are $\overrightarrow{\delta'} = (\overrightarrow{\delta}, (\delta_{n+1} \cdot a_i)_{i \in W(P)})$. It is easy to see that a semi-functional key will correctly decrypt a semi-functional ciphertext (i.e. it is nominal) if and only if $\overrightarrow{\gamma} * \overrightarrow{\delta^*} = 0 \bmod p_2$, where $\overrightarrow{\delta^*} = \left(\overrightarrow{\delta} + \left\langle 0, \cdots, 0, \sum_{i \in W(P)} \delta_{n+1} \cdot a_i \cdot ID_i \right\rangle\right)$, and assuming the identity vector $(ID_1, ID_2, \ldots, ID_j)$ *matches* the pattern $\overrightarrow{P} = (P_1, \ldots, P_j)$.

If the identity vector of the secret key, say $\overrightarrow{ID} = (ID_1, \ldots, ID_j)$, is a prefix of the challenge pattern of the ciphertext, say $\overrightarrow{P} = (P_1, \ldots, P_k)$, then the user can use the delegate algorithm to get a secret key for identity vector $\overrightarrow{ID'} = (ID_1, \ldots, ID_j, ID_{j+1}, \ldots, ID_k)$ where $ID_i = P_i$ if $P_i \neq *$ and choose randomly ID_i if $P_i = *$, $i = (j+1, \ldots, k)$.

Then the semi-functional parameters will become:

$$\overrightarrow{\gamma'} = \overrightarrow{\gamma} + \left\langle 0, \cdots, 0, -\sum_{i=j+1}^{k} \gamma_{n+2}.a_i.ID_i, 0 \right\rangle.$$

Thus, we say that this key is nominally semi-functional if $\overrightarrow{\gamma'} * \overrightarrow{\delta^*} = 0 \bmod p_2$,
where $\overrightarrow{\delta^*} = \left(\overrightarrow{\delta} + \left\langle 0, \cdots, 0, \sum_{i \in W(P)} \delta_{n+1} \cdot a_i \cdot ID_i \right\rangle\right)$.

Theorem 10 (Security of Key-Leakage Resilient $\mathsf{BBG-WIBE}$). *Under assumptions 1, 2, 3 in Appendix A and for $\ell_{SK} = (n-1-2c)\log(p_2)$, where $c > 0$ is any fixed positive constant, our key-leakage resilient $\mathsf{BBG-WIBE}$ scheme is (ℓ_{SK}) - key-leakage secure.*

The condition for c is p_2^{-c} is negligible. The length of secret key SK at level i is $(n+2+D-i)(\log(p_1)+\log(p_2)+\log(p_3))$ where D is the depth of WIBE. As we can see, the *leakage fraction* of secret key at leaf node is the biggest.

Proof. We first define several security games as follows:

- $\mathsf{KeyLeakWibe}$ game is the same as the real $\mathsf{Leak-WIBE}$ game except that all keys leaked or given to the adversary are normal key.
- $\mathsf{KeyLeakWibe}^*$ game is the same as $\mathsf{KeyLeakWibe}$ game except that in $\mathsf{KeyLeakWibe}^*$ game all **Delegate** calls are substituted by **Keyder** calls.

- $\mathsf{KeyLeakC}$ game is exactly the same as the $\mathsf{KeyLeakWibe}^*$ game except that the challenge ciphertext is semi-functional ciphertext.
- $\mathsf{KeyLeakCK}$ game is exactly the same as the $\mathsf{KeyLeakC}$ game except that all keys leaked or given to the adversary are semi-functional.
- $\mathsf{KeyLeak}_b$ game is the same as the $\mathsf{KeyLeakCK}$ game except that in one key, we call the challenge key, the adversary can access via **Create**, **Leak**, or **Reveal** queries but cannot know it is normal key or semi-functional key. The others keys the adversary can choose normal key or semi-functional key to leak or reveal, if it chooses first leakage, reveal are normal or semi all subsequent **Leak** and **Reveal** queries act on the normal or semi version. We call $\mathsf{KeyLeak}_1$ game in the case this challenge key is semi-functional key, and $\mathsf{KeyLeak}_0$ game when this challenge key is normal key.

Based on the three complexity assumptions 1, 2, and 3, we will prove the theorem by first showing that these games are indistinguishable, then prove that the adversary has no advantage in attacking the game $\mathsf{KeyLeakCK}$, .

$\mathsf{Leak}-\mathsf{WIBE} \approx \mathsf{KeyLeakWibe}$: We let q denote the number of key queries the adversary makes. For k from 0 to q, we define $Game_k$ is the same $\mathsf{Leak}-\mathsf{WIBE}$ game except that the first k keys are normal keys and the rest are semi-functional keys. $Game_0$ is $\mathsf{Leak}-\mathsf{WIBE}$ game and $Game_q$ is $\mathsf{KeyLeakWibe}$ game.

$Game_{k-1} \approx Game_k$: We will prove based on the assumption 2. From the input values of Assumption 2: $D^2 = (N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_3, g_1^z g_2^\nu, g_2^\mu g_3^\rho)$ and a challenge term T, the challenger is able to generate mpk and msk, and to answer all **Create**, **Leak**, and **Reveal** queries in both versions normal or semi. Moreover, the challenger can use T to generate the k'th key. Depending on the nature of T, the k'th is either a normal or a semi-functional key or this is either $Game_{k-1}$ or $Game_k$.

$\mathsf{KeyLeakWibe} \approx \mathsf{KeyLeakWibe}^*$: It is easy to verify that the output of the **Delegate** algorithm is identically distributed to the output of **Keygen**.

$\mathsf{KeyLeakWibe}^* \approx \mathsf{KeyLeakC}$: In $\mathsf{KeyLeakC}$, the challenge ciphertext C is semi-functional, while all keys are normal. Notice that from the input values of Assumption 1 the challenger is able to generate mpk and msk, and to answer all **Create**, **Leak**, and **Reveal** queries. Moreover, the challenger can use T to generate C and, depending on the nature of T, C can be normal as in $\mathsf{KeyLeakWibe}^*$ or semi-functional as in $\mathsf{KeyLeakC}$.

$\mathsf{KeyLeak}_0 \approx \mathsf{KeyLeak}_1$: From the input values of Assumption 2: $D^2 = (N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_3, g_1^z g_2^\nu, g_2^\mu g_3^\rho)$ and a challenge term T, the challenger is able to generate mpk and msk, and to answer all **Create**, **Leak**, and **Reveal** queries in both versions normal or semi. Moreover, the challenger can use T to generate the challenge key instead of choosing randomly bit b and gives leakage to the adversary. Depending on the nature of T, the challenger gives leakage either from a normal or from a semi-functional secret key to the adversary.

Similar to the proof of fully secure WIBE, $C_{2,i}$ in semi-functional ciphertex and E_i in semi-functional key are nominal with each other. In the case the challenge key is not capable of decrypting the challenge ciphertext, the challenger depends on the difference of advantages in the game $\mathsf{KeyLeak}_0$ and $\mathsf{KeyLeak}_1$ to determine the nature of T.

Assume that the challenge key identity vector is $ID = (ID_1, \ldots, ID_j)$, the challenge pattern is $P^* = (P_1, \ldots, P_j)$. In the case the challenge key is capable of decrypting the challenge ciphertext (adversary gets access via **Leak** queries) or if $ID_i = P_i \bmod p_2$ and $ID_i \neq P_i \bmod N$ where $i \in \overline{W}(P^*)$, the semi-functional parameters are not properly distributed. However, based on two following lemmas we get that the change in the simulator's advantage is only negligible.

In the case $ID_i = P_i \bmod p_2$ and $ID_i \neq P_i \bmod N$, we can find a non-trivial factor of N with non-negligible probability. This non-trivial factor can then be used to break Assumption 2 as in the proof of lemma 5 in [21]. If the challenge key is capable of decrypting the challenge ciphertex, the semi-functional challenge key is nominal with respect to the semi-functional challenge ciphertext.

Lemma 11. *If the assumption 2 in Appendix A holds, then for any PPT adversary $\mathcal{A}$, $\mathcal{A}$'s advantage in the* $\mathsf{KeyLeak}_b$ *game, where $b = 0$ or $b = 1$, changes only by a negligible amount if we restrict it to make queries only on the challenge identity vector, and on identity vectors such that no component* *of them is equal to a respective component from the challenge identity vector modulo p_2 and not also equal modulo N.*

Proof. The proof is similar to the proof of the lemma 5 in [21].

Lemma 12. *We suppose the leakage is at most ($\ell_{SK} = (n-1-2c)\log(p_2)$), where $c > 0$ is a fixed positive constant. Then, for any PPT adversary $\mathcal{A}$, its advantage in the* $\mathsf{KeyLeak}_1$ *game changes only by a negligible amount when the truly semi-functional challenge key is replaced by a nominal semi-functional challenge key whenever $\mathcal{A}$ declares the challenge key associated to an identity vector which matches the challenge ciphertext pattern.*

Proof. As in [20], we suppose that there exists a PPT algorithm $\mathcal{A}$ whose advantage changes by a non-negligible amount ϵ when the $\mathsf{KeyLeak}_1$ game changes as described above. Using $\mathcal{A}$, we will create a PPT algorithm $\mathcal{B}$ which will distinguish between the distributions $(\overrightarrow{\delta}, f(\tau))$ and $(\overrightarrow{\delta}, f(\tau'))$ from the corollary 6.3 in [20] with non-negligible advantage (where $m = n+1$ and $p = p_2$).

$\mathcal{B}$ simulates the game $\mathsf{KeyLeak}_1$ with $\mathcal{A}$ as follows. It starts by running the **Setup** algorithm for itself, and giving $\mathcal{A}$ the public parameters. Since $\mathcal{B}$ knows msk and generators of all the subgroups, it can make normal as well as semi-functional keys. Hence, it can respond to all $\mathcal{A}$'s non-challenge **Phase 1** queries.

With non-negligible probability, $\mathcal{A}$ must chose a challenge key in **Phase 1** with its identity vector *matches* the challenge ciphertext's pattern. (If it only did this with negligible probability, then the difference in advantages whenever it gave a *matched* identity would be negligible.) $\mathcal{B}$ will not create this challenge key, but instead will encode the leakage $\mathcal{A}$ asks for on this key in **Phase 1** as a single polynomial time computable function f with domain $\mathbb{Z}_{p_2}^{n+1}$ and with an image of size $2^{\ell_{SK}}$. It can do this by fixing the values of all other keys and fixing all other variables involved in the challenge key. $\mathcal{B}$ then receives a sample $(\overrightarrow{\delta}, f(\overrightarrow{\varGamma}))$, where $\overrightarrow{\varGamma}$ is either distributed as τ or as τ', in the notation of the corollary. $\mathcal{B}$ will use $f(\overrightarrow{\varGamma})$ to answer all of $\mathcal{A}$'s leakage queries on the challenge key by implicitly defining the challenge key as follows.

$\mathcal{B}$ chooses $r_1, r_2, z_k \in \mathbb{Z}_{p_2}$ subject to the constraint $\varGamma_{n+1} + r_1 = r_2(z_k - \sum_{i=1}^{j} a_i ID_i)$. We let g_2 denote a generator of $\mathbb{G}_2$. $\mathcal{B}$ implicitly sets the $\mathbb{G}_2$ components of the key to be $g_2^{\overrightarrow{\varGamma'}}$, where $\overrightarrow{\varGamma'}$ is defined to be

$$\overrightarrow{\varGamma'} = \left\langle \overrightarrow{\varGamma}, \underbrace{0,\ldots,0}_{D-j+1} \right\rangle + \left\langle \underbrace{0,\ldots,0}_{n}, r_1, r_2, r_2 a_{j+1}, \ldots, r_2 a_D \right\rangle$$

Note that $\overrightarrow{\varGamma}$ is of length n+1; thus r_1 is added to the last component of $\overrightarrow{\varGamma}$. $\mathcal{B}$ defines the non-$\mathbb{G}_2$ components of the key to fit their appropriate distribution.

At some point, $\mathcal{A}$ declares the pattern for the challenge ciphertext. If the challenge key had an identity vector which did not *match* the challenge ciphertext's pattern, then $\mathcal{B}$ aborts the simulation and guesses whether $\overrightarrow{\varGamma}$ is orthogonal to $\overrightarrow{\delta}$ randomly. However, the simulation continues with non-negligible probability. Suppose the challenge key's identity vector is $(ID_1, ID_2, \ldots, ID_j)$ and the challenge ciphertext's pattern is $(P_1, P_2, \ldots, P_k)$.

$\mathcal{B}$ chooses $z_c \in \mathbb{Z}_{p_2}$ subject to the constraint $\delta_{n+1} r_1 + \delta_{n+1}(z_c + \sum_{i \le j, i \in \overline{W}(P)} a_i P_i + \sum_{i \le j, i \in W(P)} a_i ID_i) r_2 = 0 \bmod p_2$. It then constructs the challenge ciphertext by using
$\overrightarrow{\delta'} = \left(\left\langle \overrightarrow{\delta}, 0 \right\rangle + \langle 0, \ldots, 0, 0, \delta_{n+1}(z_c + \sum_{i \in \overline{W}(P)} a_i P_i) \rangle, (\delta_{n+1} a_i)_{i \in W(P)} \right)$ as the challenge vector (recall that $\overrightarrow{\delta}$ is of length n+1).

Note that if $j < k$, the user chooses whatever ID_i if $P_i = *$ and chooses $ID_i = P_i$ if $P_i \neq *$, where $i = j+1, \ldots, k$, to run the delegation algorithm and get the new $\mathbb{G}_2$ components of the key:

$$\overrightarrow{\varGamma''} = \overrightarrow{\varGamma'} + \left\langle \underbrace{0,\ldots,0}_{n}, -\sum_{i=j+1}^{k} r_2 a_i ID_i, 0, \ldots, 0 \right\rangle$$

The user also runs the algorithm to recover the corresponding ciphertext by using these identities ID_i, $i = 1, \dots, k$, and now the vector $\mathbb{G}_2$ components of ciphertext is
$\overrightarrow{\delta^*} = \left(\left\langle \overrightarrow{\delta}, 0\right\rangle + \left\langle 0, \dots, 0, 0, \delta_{n+1}(z_c + \sum_{i\in\overline{W}(P)} a_i P_i + \sum_{i\in W(P)} a_i ID_i)\right\rangle\right)$.

Now, if $\overrightarrow{I}$ is orthogonal to $\overrightarrow{\delta}$, then the challenge key is nominally semi-functional (and well-distributed as such). If $\overrightarrow{I}$ is not orthogonal to $\overrightarrow{\delta}$, then the challenge key is truly semi-functional (and also well-distributed).

It is clear that $\mathcal{B}$ can easily handle **Phase 2** queries, since the challenge key cannot be queried on here when its identity vector *matches* the ciphertext's pattern. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain a non-negligible advantage in distinguishing the distributions $(\overrightarrow{\delta}, f(\tau))$ and $(\overrightarrow{\delta}, f(\tau'))$. This violates Corollary 6.3 in [20], since these distributions have a negligible statistical distance for f with this output size.

In conclusion, the challenger, depending on the difference of advantages in the game $\mathsf{KeyLeak}_0$ and $\mathsf{KeyLeak}_1$, can determine the nature of T. □

$\mathsf{KeyLeakC} \approx \mathsf{KeyLeakCK}$: we denote by Q the maximum number of queries that adversary makes. Thus, the total number of different secret keys is Q, Q is a polynomial in λ. For $q \in [0, Q]$ we define the game SF_q to be like the KeyLeakC game, semi-functional versions for the first q different keys, and normal versions for the remaining keys. The order is defined by the first leakage or reveal query made on each key. So, SF_0 is the KeyLeakC game and SF_Q is the KeyLeakCK game.
If the advantage of $\mathsf{KeyLeakC} \neq \mathsf{KeyLeakCK}$ with a non-negligible value, then there exists a $q^* \in [0, Q]$ such that the difference of advantage between two games SF_q and SF_{q+1} is non-negligible.

Assume $\mathcal{B}$ is an adversary which attacks game $\mathsf{KeyLeak}_b$, $\mathcal{B}$ simulates $\mathcal{A}$ as follow and uses the output of $\mathcal{A}$ to distinguish the difference of advantage between two games $\mathsf{KeyLeak}_0$ and $\mathsf{KeyLeak}_1$ with a non-negligible value, this is a contradiction of the result above.

$\mathcal{B}$ requests semi-functional keys for the first q^* keys, chooses the $(q^*+1)-th$ key to be the challenge key, and requests normal keys for the remaining keys. Give those to $\mathcal{A}$. If the $\mathsf{KeyLeak}_b$ challenger picked $b = 0$, then $\mathcal{A}$ plays the SF_{q^*} game. Otherwise, it plays the SF_{q^*+1} game.

KeyLeakCK gives no advantage to the adversary: we use $\mathcal{A}$ with non-negligible advantage in breaking KeyLeakCK game to build a PPT simulator $\mathcal{B}$ that breaks assumption 3. From the input of the assumption's challenger, $D^3 = (N, \mathbb{G}, \mathbb{G}_T, e, g_1, g_2, g_3, g_1^{\alpha} g_2^{\nu}, g_1^{z} g_2^{\mu})$ and T which is either $e(g_1, g_1)^{\alpha z}$ or a random term of $\mathbb{G}_T$, $\mathcal{B}$ can answer all queries from $\mathcal{A}$. When $\mathcal{A}$ gives the challenge key to $\mathcal{B}$, $\mathcal{B}$ uses T to create the ciphertext. Depending on the nature of T, this is a ciphertext of real message or ciphertext of random message. If this is a ciphertext of real message then $\mathcal{B}$ stimulates the KeyLeakCK game.

All in all, from the above reductions between the successive games, we deduce that $\mathsf{Leak} - \mathsf{WIBE}$ $\approx$ KeyLeakCK and therefore, the advantage of adversary in $\mathsf{Leak} - \mathsf{WIBE}$ game is negligible. □

3.3 Key-Leakage Resilient Revoke Scheme Immune to Pirates 2.0

The definition and adaptive security model of KIDTR scheme can be found in Appendix D.1 and D.2. The construction of KIDTR is the same as in [27] except we use KWIBE instead of WIBE for encryption. The construction and security of KIDTR are provided in Appendix D.3 and 21.

Proposition 13. *In* KIDTR *scheme, if we call p_1, p_2, p_3 are primes of $\lambda_1, \lambda_2, \lambda_3$ bits, then each user's secret key with length $m = (n+2)(\lambda_1 + \lambda_2 + \lambda_3)$ is m'-independent source where $m' = ((n+1)(\lambda_1 + \lambda_2 + \lambda_3) + \lambda_2 + \lambda_3)$.*

Proof. In our KIDTR scheme, we make use of a KWIBE scheme in which each user's secret key is at leaf node 3.2, therefore an user's secret key is of the following form:

$$SK = \overrightarrow{K_1} = \left(\left\langle g_1^{t_1}, g_1^{t_2}, \dots, g_1^{t_n}, g_1^{\alpha}\left(h \cdot \prod_{i=1}^{j} u_i^{ID_i}\right)^{-r} \prod_{i=1}^{n} g_1^{-x_i t_i}, g_1^{r}\right\rangle * g_3^{\overrightarrow{\rho}} * g_2^{\overrightarrow{\gamma}}\right)$$

where $r, t_1, t_2, \ldots, t_n, z_k \overset{\$}{\leftarrow} \mathbb{Z}_N$, $\overrightarrow{\rho} \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$, and $\overrightarrow{\gamma} = (\gamma_1, \ldots, \gamma_{n+2})$ in which $(\gamma_1, \ldots, \gamma_n, \gamma_{n+2}) \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}$, $\gamma_{n+1} = \gamma_{n+2}(z_k - \sum_{i=1}^{j} a_i ID_i)$.
We realize that in each secret key, the elements corresponding to the indices $1, \ldots, n, n+2$ are randomly generated in the whole group $\mathbb{G} = \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_3$, the element corresponding to the indice $n+1$ is not independent in $\mathbb{G}_1$ but randomly generated in $\mathbb{G}_2 \times \mathbb{G}_3$. Therefore, it's easy to see that each user's secret key is of $(n+2)(\lambda_1 + \lambda_2 + \lambda_3)$ bit length and is a $((n+1)(\lambda_1 + \lambda_2 + \lambda_3) + \lambda_2 + \lambda_3)$-independent source.

Theorem 14. *The* KIDTR *scheme is immune to Pirates 2.0 attacks for any choice of parameters* $n, c, \lambda_1, \lambda_2$ *such that* $2^{(n-1-2c)\lambda_2 - \lambda_1} > N_u$*, where* N_u *is the number of subscribed users in the systems*

Proof. From the theorems 10 and theorem 21, we decude that the KIDTR scheme is $\ell_{SK}-$leakage resilient with $\ell_{SK} = (n-1-2c)\lambda_2$ for any fixed positive constant $c > 0$ (such that p_2^{-c} is negligible). From the theorem 7, one cannot mount a Pirates 2.0 attack with an anonymity level larger than $\alpha = \frac{N_u}{2^{\ell_{SK}+m'-m}} = \frac{N_u}{2^{(n-1-2c)\lambda_2-\lambda_1}} < 1$. □

We note that there is no need to choose particular parameters for our system. For example, simply with $c = 1, n = 5$ and $\lambda_1 = \lambda_2 = 512$ ($p_2^{-c} = 2^{-512}$ is negligible) and suppose that there are $N_u = 2^{40}$ subscribed users, our system is immune to Pirates 2.0 because $2^{(n-1-2c)\lambda_2-\lambda_1} = 2^{512} > N_u$ and the user's secret key contains only 7 elements in $\mathbb{G}$.

4 Discussion

4.1 Traceability in our scheme

The effectiveness of Pirates 2.0 in practice is to allow a very large scale of public collaboration of traitors. This relies on the anonymity of each contributor. By formally proving in Section 2 that the anonymity can not be assured, there is no risk for a large scale of public collaboration of traitors in our system. Concerning to the classical tracing where traitors contribute their whole secret keys, because our scheme which is based on the structure of complete-subtree scheme, it achieves the same level of traceability as the schemes in the subset-cover framework [23]: the tracer, having black-box access to a pirate decryption box $\mathbb{D}$, can outputs either a set of traitors or a way to render the illegal decryption box useless.

4.2 Computational entropy and Pirates 2.0

We consider the information-theoretic notion of entropy and design a scheme where the keys of users are all high-independent source. One might ask a natural question if it suffices to use the computational entropy (a distribution X has k bit computational entropy if there is a distribution Y of k bit min-entropy such that X an Y and computationally indistinguishable). A positive result would imply that almost all known algebraic broadcast encryptions resist Pirates 2.0 attacks if they are key-leakage resilient. Unfortunately, it seems hard for us that the computational entropy is suitable in the context of Pirates 2.0. The main reason is that if an user has a key of k bit computational entropy, the key can still remain k bit computational entropy even after the user contributes some k' bit information about the key. Therefore, we cannot control the remaining computational entropy of the keys after each round of leaking information in Pirates 2.0, especially when the users choose the form to leak information. It seems thus an open problem to determine whether the other known broadcast encryptions resist Pirates 2.0. As a concrete example, in BGW scheme[7], the key of the user i is $d_i = v^{\alpha^i}$ which is zero-independent source (one key is totally determined in information-theoretic sense by an another key). These keys have high computational entropy (under the bilinear Diffie-Hellman Exponent assumption) but as explained above, it's not easy to explore this computational entropy in the context of Pirates 2.0.

4.3 Active leakage in cryptography

In the last few years, theoretical foundations have been developed in order to formally address the problem of side-channel attacks - a very frequent and practical attack against implementations of cryptographic protocols. These led to the development of Leakage Resilient Cryptography with the objective is to deal with any form of side-channel attacks. The source of leakage comes from the possibility of an adversary to extract the information about the secret key. We would like to furthermore investigate the question of active leakage where users intentionally leak partial information of their secret keys. The main property is probably the anonymity of the colluded users: users want to leak information in discretion to break the security of the system. This scenario could be very relevant in multi-user cryptography. In fact, Pirates 2.0 exactly formalizes the active leakage in the context of multi-user encryption and the view of Pirates 2.0 as a form of leakage resilience led us to the research in this paper. We believe that the question of active leakage is deserved to be more studied in many scenarios of multi-user cryptography including secret sharing, threshold cryptography. As an example, we wonder whether there exist a threshold scheme that is secure against classical collusions of less than t users but is vulnerable to a collusion of more than t users in the active leakage model where the main requirement is that all the contributors rest assured to be anonymous even against an unbounded authority.

References

1. M. Abdalla, J. Birkett, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven, J. C. N. Schuldt, and N. P. Smart. Wildcarded identity-based encryption. *Journal of Cryptology*, 24(1):42–82, Jan. 2011.
2. M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, and N. Smart. Identity-based encryption gone wild. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 300–311. Springer, July 2006.
3. M. Abdalla, A. W. Dent, J. Malone-Lee, G. Neven, D. H. Phan, and N. P. Smart. Identity-based traitor tracing. In T. Okamoto and X. Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 361–376. Springer, Apr. 2007.
4. J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, Aug. 2009.
5. S. Berkovits. How to broadcast a secret (rump session). In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 535–541. Springer, Apr. 1991.
6. O. Billet and D. H. Phan. Traitors collaborating in public: Pirates 2.0. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 189–205. Springer, Apr. 2009.
7. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, Aug. 2005.
8. Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st Annual Symposium on Foundations of Computer Science*, pages 501–510. IEEE Computer Society Press, 2010.
9. D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. J. Lipton, and S. Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In S. P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 479–498. Springer, Feb. 2007.
10. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, Aug. 1994.
11. P. D'Arco and A. L. P. del Pozo. Fighting Pirates 2.0. In *Proc. of the 9th International Conference on Applied Cryptography and Network Security —ACNS 2011*, Lecture Notes in Computer Science. Springer, 2011.
12. G. Di Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In S. Halevi and T. Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 225–244. Springer, Mar. 2006.
13. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer, May 2004.
14. S. Dziembowski and K. Pietrzak. Intrusion-resilient secret sharing. In *48th Annual Symposium on Foundations of Computer Science*, pages 227–237. IEEE Computer Society Press, Oct. 2007.
15. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302. IEEE Computer Society Press, Oct. 2008.

16. A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, Aug. 1994.
17. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, Aug. 2003.
18. J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, Dec. 2009.
19. A. Kiayias and S. Pehlivanoglu. Pirate evolution: How to make the most of your traitor keys. In A. Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 448–465. Springer, Aug. 2007.
20. A. B. Lewko, Y. Rouselakis, and B. Waters. Achieving leakage resilience through dual system encryption. In Y. Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 70–88. Springer, Mar. 2011.
21. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In D. Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, Feb. 2010.
22. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In M. Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, Feb. 2004.
23. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, Aug. 2001.
24. M. Naor and B. Pinkas. Efficient trace and revoke schemes. In Y. Frankel, editor, *FC 2000: 4th International Conference on Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Feb. 2000.
25. M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, Aug. 2009.
26. C. Petit, F.-X. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In M. Abe and V. Gligor, editors, *ASIACCS 08: 3rd Conference on Computer and Communications Security*, pages 56–65. ACM Press, Mar. 2008.
27. D. H. Phan and V. C. Trinh. Identity-based trace and revoke schemes. In X. Boyen and X. Chen, editors, *ProvSec 2011: 5th International Conference on Provable Security*, volume 6980 of *Lecture Notes in Computer Science*, pages 204–221. Springer, Oct. 2011.
28. F.-X. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, Apr. 2009.
29. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, Aug. 2009.
30. X. Zhao and F. Zhang. Traitor tracing against public collaboration (full version). In *ISPEC '11: The 7th International Conference on Information Security Practice and Experience*, Guangzhou / China, 2011.

A Composite Order Bilinear Groups

We recall three assumptions from [21].

Assumption 1 (Subgroup decision problem for 3 primes) Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \overset{\$}{\leftarrow} \mathcal{G}; g \overset{\$}{\leftarrow} G_{p_1}; X_3 \overset{\$}{\leftarrow} G_{p_3}.$$

$$D = (\mathbb{G}, g, X_3); \quad T_1 \overset{\$}{\leftarrow} G_{p_1p_2}, T_2 \overset{\$}{\leftarrow} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := \mid Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \mid .$$

We note that T_1 can be written (uniquely) as the product of an element of G_{p_1} and an element of G_{p_2}. We refer to these elements as the "G_{p_1} part of T_1" and the "G_{p_2} part of T_1" respectively.

Definition 15. *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 2 Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{\$} \mathcal{G}; g, X_1 \xleftarrow{\$} G_{p_1}; X_2, Y_2 \xleftarrow{\$} G_{p_2}; X_3, Y_3 \xleftarrow{\$} G_{p_3}.$$

$$D = (\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3); \quad T_1 \xleftarrow{\$} G, T_2 \xleftarrow{\$} G_{p_1p_3}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) :=\mid Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \mid .$$

We use $G_{p_1p_3}$ to denote the subgroup of order p_1p_3 in G. We note that T_1 can be (uniquely) written as the product of an element of G_{p_1}, an element of G_{p_2}, and an element of G_{p_3}. We refer to these as the "G_{p_1} part of T_1", the "G_{p_2} part of T_1", and the "G_{p_3} part of T_1", respectively. T_2 can similarly be written as the product of an element of G_{p_1} and an element of G_{p_3}.

Definition 16. *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 3 Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{\$} \mathcal{G}; \alpha, s \xleftarrow{\$} Z_N; g \xleftarrow{\$} G_{p_1}; X_2, Y_2, Z_2 \xleftarrow{\$} G_{p_2}; X_3 \xleftarrow{\$} G_{p_3}.$$

$$D = (\mathbb{G}, g, g^\alpha X_2, X_3, g^s Y_2, Z_2); \quad T_1 = e(g, g)^{\alpha s}, T_2 \xleftarrow{\$} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) :=\mid Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \mid .$$

Definition 17. *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

B Construction of BBG-WIBE Scheme in Composite Order Groups

Setup The setup algorithm chooses a bilinear group $\mathbb{G}$ of order $N = p_1p_2p_3$. We will assume that users are associated with vectors of identities whose components are elements of $\mathbb{Z}_N$. We let D denote the maximum depth of the WIBE, the setup algorithm chooses a generator $g \in \mathbb{G}_1$, $X_3 \in \mathbb{G}_3$, $\alpha, b, a_1, \ldots, a_D \xleftarrow{\$} \mathbb{Z}_N$. Denote $u_1 = g^{a_1}, \ldots, u_D = g^{a_D}, h = g^b$. The master key is $\mathsf{msk} = \alpha$, the public parameters are published as:

$$\mathsf{mpk} = (N, g, h, u_1, \ldots, u_D, X_3, e(g, g)^\alpha)$$

Keyder$(\mathsf{msk}, (ID_1, ID_2, \ldots, ID_j), \mathsf{mpk})$ The key generation algorithm chooses $r \xleftarrow{\$} \mathbb{Z}_N$ and also chooses random elements $R_3, R_3', R_{j+1}, \ldots, R_D$ of $\mathbb{G}_3$. It sets:

$$K_1 = g^r R_3, K_2 = g^\alpha \left(u_1^{ID_1} \cdots u_j^{ID_j} h\right)^r R_3', E_{j+1} = u_{j+1}^r R_{j+1}, \ldots, E_D = u_D^r R_D.$$

Delegate Given a key $K_1', K_2', E_{j+1}', \ldots, E_D'$ for $(ID_1, \ldots, ID_j)$, the delegation algorithm creates a key for $(ID_1, \ldots, ID_{j+1})$ as follows. It chooses $r' \xleftarrow{\$} \mathbb{Z}_N$ and random elements of $\mathbb{G}_3$ denoted, e.g., by $\tilde{R}_3$. The new key is set as:

$$K_1 = K_1' g^{r'} \tilde{R}_3, \quad K_2 = K_2' \left(u_1^{ID_1} \cdots u_j^{ID_j} h\right)^{r'} (E_{j+1}')^{ID_{j+1}} u_{j+1}^{r' ID_{j+1}} \tilde{R}_3',$$
$$E_{j+2} = E_{j+2}' u_{j+2}^{r'} \tilde{R}_{j+2}, \ldots, E_D = E_D' u_D^{r'} \tilde{R}_D$$

We note that this new key is fully re-randomized: its only tie to the previous key is in the values $(ID_1, \ldots, ID_j)$.

Enc$(M, (P_1, P_2, \ldots, P_j))$ The encryption algorithm chooses $s \xleftarrow{\$} \mathbb{Z}_N$ and outputs the ciphertext:

$$C_0 = M \cdot e(g,g)^{\alpha s}, C_1 = (h \cdot \prod_{i \in \overline{W}(P)} u_i^{P_i})^s, C_2 = g^s, C_3 = (C_{3,i} = u_i^s)_{i \in W(P)}$$

Dec Any other receiver with identity $ID = (ID_1, ID_2, \ldots, ID_j)$ matching the pattern P to which the ciphertext was created can decrypt the ciphertext as follows
First, he recovers the message by computing

$$C_1' = C_1 \cdot \prod_{i \in W(P)} (u_i^s)^{ID_i}$$

then computes the blinding factor as:

$$\frac{e(K_2, C_2)}{e(K_1, C_1')} = \frac{e(g,g)^{\alpha s} e(u_1^{ID_1} \cdots u_j^{ID_j} h, g)^{rs}}{e(g, u_1^{ID_1} \cdots u_j^{ID_j} h)^{rs}} = e(g,g)^{\alpha s}$$

B.1 Security of BBG-WIBE Scheme in Composite Order Groups

Semi-functional Keys. We let g_2 denote a generator of $\mathbb{G}_{p_2}$. To create a semi-functional key for identity $(ID_1, \ldots, ID_j)$, we first create a normal key $K_1', K_2', E_{j+1}', \ldots, E_D'$ using the key generation algorithm. We choose random exponents $\gamma, z_k \xleftarrow{\$} \mathbb{Z}_N$ and output

$$K_1 = K_1' g_2^{\gamma}, K_2 = K_2' g_2^{\gamma(z_k + \sum_{i=1}^{j} a_i ID_i)}, E_{j+1} = E_{j+1}' g_2^{\gamma a_{j+1}}, \ldots, E_D = E_D' g_2^{\gamma a_D},$$

Semi-functional Ciphertext. A semi-functional ciphertext is created for pattern $(P_1, \ldots, P_j)$ as follows: first, we use the encryption algorithm to form a normal ciphertext C_0', C_1', C_2', C_3'. We choose random exponents $x, z_c \in \mathbb{Z}_N$ and output:

$$C_0 = C_0', C_1 = C_1' g_2^{x(z_c + \sum_{i \in \overline{W}(P)} a_i P_i)}, C_2 = C_2' g_2^x, C_3 = (C_{3,i} = C_{3,i}' \cdot g_2^{x a_i})_{i \in W(P)}$$

We note that when a semi-functional key is used to decrypt a semi-functional ciphertext, the decryption algorithm will compute the blinding factor multiplied by the additional term of $e(g_2, g_2)^{x\gamma(z_k + \sum_{i=1}^{j} a_i ID_i - z_c - \sum_{i \in \overline{W}(P)} a_i P_i - \sum_{i \in W(P)} a_i ID_i)}$. If the identity $(ID_1, \ldots, ID_j)$ matches the pattern $(P_1, \ldots, P_j)$ and $z_k = z_c$, decryption will still work. In this case, the key is nominally semi-functional.
We recall three assumptions 1, 2, 3 in Appendix A.

Theorem 18. *If Assumptions 1, 2, and 3 hold, then our* WIBE *system is fully secure.*

Overview of Proof of Security Our proof of security will be structured as a hybrid argument over a sequence of games. The first game, *GameReal*, is the real WIBE security game. The next game, *GameReal'*, is the same as the real game except that all key queries will be answered by fresh calls to the key generation algorithm (the challenger will not be asked to delegate keys in a particular way). The next game, *GameRestricted* is the same as *GameReal'* except that the attacker cannot ask for keys for identities which have at least a component is equal to a respective component of the challenge pattern (at positions not a wildcard) modulo p_2 and not also equal modulo N. We will retain this restriction in all subsequent games. We let q denote the number of key queries the attacker makes. For k from 0 to q, we define $Game_k$ as:

$Game_k$ This is like *GameRestricted*, except that the ciphertext given to the attacker is semi-functional and the first k keys are semi-functional. The rest of the keys are normal. In $Game_0$, only the challenge ciphertext is semi-functional. In $Game_q$, the challenge ciphertext and all of the keys are semi-functional. We define *GameFinal* to be like $Game_q$, except that the challenge ciphertext is a semi-functional encryption of a random message, not one of the messages provided by the attacker. We will prove the security of the scheme by showing these games are indistinguishable. Informally, we have:

GameReal $\approx$ *GameReal*$'$: Keys are identically distributed whether they are produced by the key delegation algorithm from a previous key or from a fresh call to the key generation algorithm. Thus, in the attacker's view, there is no difference between these games.

GameReal$'$ $\approx$ *GameRestricted*: Essentially, if the adversary is able to ask for key for identities which have at least a component is equal to a respective component of the challenge pattern (at positions not a wildcard) modulo p_2 and not also equal modulo N, then this means that the adversary can find a non-trivial factor of N and can be used to break the Assumption 2 (the same proof of lemma 5 in [21]).

GameRestricted $\approx$ $Game_0$: In $Game_0$, the challenge ciphertext C is semi-functional, while all keys are normal. Notice that from the input values of Assumption 1 the challenger is able to generate mpk and msk, and to answer to all secret key queries. Moreover, the challenger can use T to generate C and, depending on the nature of T, C can be normal as in *GameRestricted* or semi-functional as in $Game_0$.

$Game_{k-1} \approx Game_k$: Under Assumption 2, these two games are indistinguishable. From the input values $(g, X_1X_2, X_3, Y_2Y_3, T)$ of Assumption 2, the challenger is able to generate mpk and msk by using g and X_3. The challenger can answer the first $k-1$ secret key queries, which are semi-functional, by employing Y_2Y_3, g, X_3. The last $q-k$ queries, which are normal, can be answered by invoking the key generation algorithm using msk. Finally, the challenger can generate the ciphertext by employing X_1X_2 and generate the $k-th$ secret key by employing T.
Now, if $T \in G_{p_1p_3}$, then the $k-th$ secret key is normal and the joint distribution of the $k-th$ secret key and the challenge ciphertext is as in $Game_{k-1}$. In contrast, if $T \in G_{p_1p_2p_3}$, then the joint distribution of the $k-th$ secret key and the challenge ciphertext is as in $Game_k$, and the $k-th$ key is nominally semi-functional with respect to the challenge ciphertext. Hence, the simulator cannot test by himself the nature of T. Moreover, the nominality of $k-th$ key is hidden to the adversary under the restriction of the game that the adversary cannot ask secret keys for identities matching with the challenge patterns, and under the restriction of *GameRestricted.* The nominality of semi components $C_{3,i}$ and semi components E_i is also hidden to the simulator and the adversary under the choosing compatibly the distribution of exponents of G_{p_2} components in the semi-functional key and the semi-functional ciphertext

$Game_q \approx GameFinal$: In $Game_q$, the challenge ciphertext and secret keys are semi-functional. It is easy to see that these two games are indistinguishable under Assumption 3.

$GameFinal$ gives no advantage: From the input of the assumption 3, (N, $\mathbb{G}$, $\mathbb{G}_T$, e, g_1, g_2, g_3, $g_1^{\alpha}g_2^{\nu}$, $g_1^{z}g_2^{\mu}$) and T which is either $e(g_1, g_1)^{\alpha z}$ or a random term of $\mathbb{G}_T$, challenger can answer all queries. When the challenger receives the challenge key it uses T to create the ciphertext. Depending on the nature of T, this is a ciphertext of real message or ciphertext of random message. If this is a ciphertext of real message then challenger stimulates the $GameFinal$ game. Hence the attacker can obtain no advantage in breaking the scheme.

C KWIBE Scheme

Security Model for KWIBE

Formally, the security model of a ℓ_{SK}-key-leakage resilient WIBE, we call Leak $-$ WIBE security game, is defined as follows:
We let I^* denote the set of all possible identity vectors, $\mathcal{R}$ denote the set of all revealed identities

Setup : The challenger makes a call to **Setup**(1^λ) and gets the master secret key msk and the public parameters mpk. It gives mpk to the attacker. Also, it sets $\mathcal{R} = \emptyset$ and $\mathcal{T} = \emptyset$, note that $\mathcal{R} \subseteq I^*, \mathcal{T} \subseteq \{I^*, \mathcal{SK}, \mathcal{N}\}$ (identity vectors - secret keys - leaked bits) thus initially no leakage on each secret key.

Phase 1 : The adversary can be interleaved in any possible way to request three types of query:

Create($\overrightarrow{I}$): The challenger initially scans $\mathcal{T}$ to find the identity vector $\overrightarrow{I}$. If this identity vector exists in $\mathcal{T}$, it responds with $\perp$.
Otherwise, the challenger makes a call to **KeyDer**(msk, $\overrightarrow{I}$) $\rightarrow SK_I$ and adds the tuple ($\overrightarrow{I}$, SK_I, 0) to the set $\mathcal{T}$.

Leak ($\overrightarrow{I}$, f): In this query, the adversary requests leakage from a key that has identity $\overrightarrow{I}$ with a polynomial-time computable function f of constant output size. The challenger scans $\mathcal{T}$ to find the specified identity vector. It is of the form $(\overrightarrow{I}, SK_I, L)$. It checks if $L+ \mid f(SK_I)| \leq \ell_{SK}$. If this is true, it responds with $f(SK_I)$ and updates the L in the tuple with $L+| f(SK_I)|$. If the checks fails, it returns $\perp$ to the adversary.

Reveal ($\overrightarrow{I}$): Now the adversary requests the entire key with identity vector $\overrightarrow{I}$. The challenger scans $\mathcal{T}$ to find the requested entry. Let's say the tuple is ($\overrightarrow{I}$, SK_I, L). The challenger responds with SK_I and adds the identity vector $\overrightarrow{I}$ to the set $\mathcal{R}$.

Challenge : The adversary submits a challenge pattern $\overrightarrow{P^*}$ with the restriction that no identity vector in $\mathcal{R}$ *matches* $\overrightarrow{P^*}$. It also submits two messages M_0, M_1 of equal size. The challenger flips a uniform coin $c \xleftarrow{\$} \{0,1\}$ and encrypts M_c under $\overrightarrow{P^*}$ with a call to **Enc**(M_c, $\overrightarrow{P^*}$). It sends the resulting ciphertext CT^* to the adversary.

Phase 2 : This is the same as **Phase 1**, except the only allowed queries are **Create** queries for all identity vector, and **Reveal** queries for secret keys with identity vectors which do not *matches* $\overrightarrow{P^*}$.

Guess : The adversary outputs a bit $c' \xleftarrow{\$} \{0,1\}$. We say it succeeds if $c' = c$.

Definition 19. *A* KWIBE *scheme is* (ℓ_{SK})*-key-leakage secure if all PPT adversaries have at most a negligible advantage in the above security game.*

D KIDTR Scheme

D.1 Definition

We follow the same framework of the identity-based traitor tracing (IBTT) in [3] and the identity-based trace and revoke (IDTR) in [27]. Under this framework, each group is associated with an identity string $ID \in \{0,1\}^*$. The maximum number in each group is assumed to be bounded by $N_u = 2^l$. Each user in a group is associated with an index $id \in \{0,1\}^l$ and is provided a personal decryption key $d_{ID,id}$. Let $\mathcal{N}_{ID}$ be the set of all users in the group ID and $\mathcal{R}_{ID}$ be a set of revoked users, the system should be able to to allow anyone to encrypt a message to the group ID such that any user $u \in \mathcal{N}_{ID} \backslash \mathcal{R}_{ID}$ can correctly decrypt the ciphertexts, while the coalition of all members of $\mathcal{R}_{ID}$ cannot correctly decrypt.

Formally, a key-leakage resilient IDTR scheme consists of five polynomial-time algorithms (**Setup**, **KeyDer**, **Enc**, **Dec**, **Trace**):

Setup($1^k, N_u$)**:** The key generation algorithm taking as input security parameter 1^k and number of users for each group N_u (we assume that the maximum number of users in each group is bounded by N_u). This algorithm generates a master public key mpk, a master secret key msk.

KeyDer(msk, ID, id)**:** The key extraction algorithm which given the master secret key msk, a group identity $ID \in \{0,1\}^*$ and a user identity id generates a user secret key $d_{ID,id}$.

Enc(mpk, $ID, \mathcal{R}_{ID}, M$)**:** The encryption algorithm which on input of the master public key mpk, a group identity ID, a revocation list $\mathcal{R}_{ID}$ of revoked users in the group ID, and a message M outputs a ciphertext C.

Dec($d_{ID,id}, C$)**:** The decryption algorithm which on input of a user secret key $d_{ID,id}$ and a ciphertext C outputs a plaintext message M, or $\perp$ to indicate a decryption error.
For correctness we require that **Dec**($d_{ID,id}$, **Enc**(mpk, $ID, \mathcal{R}_{ID}, M$)) $= M$ with probability one for all $k \in \mathbb{N}$, $ID, M \in \{0,1\}^*$, $id \in \{0,1\}^l$, (mpk, msk) $\xleftarrow{\$}$ **Setup**($1^k, N_u$) and $d_{ID,id} \xleftarrow{\$}$ **KeyDer**(msk, ID, id).

Trace$^{\mathbb{D}}(\mathsf{msk}, ID)$: The traitor tracing algorithm which has oracle access to a "pirate" decryption box $\mathbb{D}$. The tracing algorithm takes as input the master secret key msk and a group identity ID, and outputs either a set of user identifiers (called "traitors") $T \subset \mathcal{N}_{ID}$ or a way to render the illegal decryption box useless.

D.2 Security Model for Key-Leakage Resilient Identity-Based Trace and Revoke Systems.

Setup: The challenger takes a parameter k, a maximum number of users in each group N_u and runs $\mathit{setup}(1^k, N_u)$ algorithm. The master public key mpk is passed to the adversary. Also, it sets $\mathcal{R}_{ID} = \emptyset, \mathcal{T}_{ID} = \emptyset$, note that $\mathcal{R}_{ID} \subseteq \langle ID, id\rangle$, and $\mathcal{T}_{ID} \subseteq (\langle ID, id\rangle, \mathcal{SK}, \mathcal{N})$ (group's identity and users' identities - secret key of users - leaked bits) for all ID. Thus initially has no leakage on each secret key.

Phase 1: The adversary can be interleaved in any possible way to request three types of query:

1. **Create**(ID, id): The challenger initially scans $\mathcal{T}_{ID}$ to find the identity (ID, id). If this identity exists in $\mathcal{T}_{ID}$, it responds with $\perp$.
   Otherwise, the challenger makes a call to **KeyDer**$(\mathsf{msk}, ID, id) \to d_{ID,id}$ and adds the tuple $((ID, id),\ d_{ID,id},\ 0)$ to the set $\mathcal{T}_{ID}$.
2. **Leak**$((ID, id),\ f)$ In this query, the adversary requests leakage from a key that has identity (ID, id) with a polynomial-time computable function f of constant output size. The challenger scans $\mathcal{T}_{ID}$ to find the specified identity. It is of the form $((ID, id),\ d_{ID,id},\ L)$. It checks if $L + \mid f(d_{ID,id})| \leq \ell_{SK}$. If this is true, it responds with $f(d_{ID,id})$ and updates the L in the tuple with $L + \mid f(d_{ID,id})|$. If the checks fails, it returns $\perp$ to the adversary.
3. **Reveal**(ID, id): Now the adversary requests the entire key with identity (ID, id). The challenger scans $\mathcal{T}_{ID}$ to find the requested entry. Let's say the tuple is $((ID, id),\ d_{ID,id},\ L)$. The challenger responds with $d_{ID,id}$ and adds the identity (ID, id) to the set $\mathcal{R}_{ID}$.

Challenge: The adversary submits two equal length messages M_0, M_1 and an identity ID^*. The challenger picks a random bit $b \in \{0, 1\}$ and set $C = \text{Encrypt}(msk, ID^*, \mathcal{R}_{ID^*}, M_b)$. The ciphertext C is passed to the adversary.

Phase 2: This is identical to phase 1 except that the allowed queries are **Create** queries, and only **Reveal**(ID, id) queries in which $ID \neq ID^*$ or $ID = ID^*$ and $id \in \mathcal{R}_{ID^*}$.

Guess: The adversary outputs a guess b' of b.

Definition 20. *A* KIDTR *scheme is (ℓ_{SK})-key leakage secure if all PPT adversaries have at most a negligible advantage in the above security game.*

D.3 Generic Construction of KIDTR

The construction of KIDTR closely follows the construction of WIBE-IDTR in [27], using the new primitive KWIBE instead of WIBE for encryption. We integrate KWIBE into the complete subtree method: each group $ID \in \{0, 1\}^*$ represents a binary tree and each user $id \in \{0, 1\}^l$ ($id = id_1 id_2 \cdots id_l$, $id_i \in \{0, 1\}$) in a group ID is assigned to be a leaf of the binary tree rooted at ID. For encryption, we will use a KWIBE of depth $l + 1$, each user is associated with a vector $(ID, id_1, \cdots, id_l)$.

Setup$(1^k, N_u)$**:** Take a security parameter k and the maximum number in each group N_u (thus $l = \lceil \log_2 N_u \rceil$). Run the setup algorithm of KWIBE with the security parameter k and the hierarchical depth $L = l + 1$ which returns $(\mathsf{mpk}, \mathsf{msk})$. The setup then outputs $(\mathsf{mpk}, \mathsf{msk})$. As in the complete subtree method, the setup also defines a data encapsulation method $E_K : \{0, 1\}^* \to \{0, 1\}^*$ and its corresponding decapsulation D_K.

Keyder(msk, ID, id)**:** Run the key derivation of KWIBE for $l{+}1$ level identity $WID = (ID, id_1, \ldots, id_l)$ (the j-th component corresponds to the j-th bit of the identity id) and get the decryption key d_{WID}. Output $d_{ID,id} = d_{WID}$.

Enc$(\mathsf{mpk}, ID, \mathcal{R}_{ID}, M)$**:** A sender wants to send a message M to a group ID with the revocation list $\mathcal{R}_{ID}$. The revocation works as in the complete subtree scheme. Considering a group ID with its revocation list $\mathcal{R}_{ID}$, the users in $\mathcal{N}_{ID}\backslash\mathcal{R}_{ID}$ are partitioned into disjoint subsets $S_{i_1}, \ldots, S_{i_w}$ which are all the subtrees of the original tree (rooted at ID) that hang off the Steiner tree defined by the set $\mathcal{R}_{ID}$.
Each subset S_{i_j}, $1 \leq j \leq w$, is associated to an l+1 vector identity $ID_{S_{i_j}} = (ID, id_{i_j,1}, \ldots, id_{i_j,k}, *, .., *)$ where $id_{i_j,1}, \ldots, id_{i_j,k}$ is the path from the root ID to the node S_{i_j} and the number of wildcards $*$ is $l - k$. The encryption algorithm randomly chooses a session key K, encrypts M under the key K by using a symmetric encryption, and outputs as a header the encryption of KWIBE for each $ID_{S_{i_1}}, \ldots, ID_{S_{i_w}}$.

$$C = \langle [i_1, \ldots, i_w][\mathsf{KWIBE}.\mathbf{Enc}(\mathsf{mpk}, ID_{S_{i_1}}, K), \ldots, \mathsf{KWIBE}.\mathbf{Enc}(\mathsf{mpk}, ID_{S_{i_w}}, K)], E_K(M)\rangle$$

Dec$(d_{ID,id}, C)$**:** The user received the ciphertext C as above. First, find j such that $id \in S_{i_j}$ (in case $id \in \mathcal{R}_{ID}$ the result is null). Second, use private key $d_{ID,id}$ to decrypt $\mathsf{KWIBE}.\mathbf{Enc}(\mathsf{mpk}, ID_{S_{i_j}}, K)$ to obtain K. Finally, compute $D_K(E_K(M))$ to recover the message M.

Trace$^{\mathbb{D}}(\mathsf{msk}, ID)$**:** Tracing algorithm takes as input msk, ID, an illegal decryption box $\mathbb{D}$, returns either a subset consisting at least one traitor or a new partition of $\mathcal{N}_{ID}\backslash\mathcal{R}_{ID}$ that renders the illegal decryption box useless.

D.4 Proof of Security of KIDTR Scheme

Theorem 21 (Security of KIDTR**).** *If the* KWIBE *is* (ℓ_{SK}) *- key-leakage secure then our* KIDTR *is also* (ℓ_{SK}) *- key-leakage secure.*

Proof. Our proof follows closely to the proof of theorem 2 in [27]. We also organize our proof as a sequence of games. The first game **Game 0** defined will be the real KIDTR game and the last one will be one in which the adversary has no advantage unconditionally. We will show that each game is indistinguishable from the next, under the assumptions of the security of KWIBE.

Game 0: This is the real attack game of an adversary $\mathcal{B}$ against the proposed KIDTR system. After receiving the public key mpk, $\mathcal{B}$ can issue adaptively three types of query **Create**, **Leak**, and **Reveal** on identity (ID, id). The challenger can easily responds these queries.
$\mathcal{B}$ finally outputs two equal length plantexts $M_0, M_1 \in \mathcal{M}$ and a targeted set ID^*.
The revoked set $\mathcal{R}_{\mathcal{ID}^*}$ consists the users' identity id such that (ID^*, id) has been asked in the **Reveal** query by adversary $\mathcal{B}$.
The challenger picks then a random bit $b \in \{0, 1\}$ and set $C = \mathsf{KIDTR.Enc}(msk, \mathcal{N}_{ID^*}\backslash\mathcal{R}_{ID^*}, M_b)$. It sends C as the challenge to $\mathcal{B}$.
Upon receiving the challenge C, $\mathcal{B}$ outputs a guess $b' \in \{0, 1\}$. $\mathcal{B}$ wins the game if $b' = b$.
In our construction, the encryption of trace and revoke system is performed as:

$$\begin{aligned}&\mathsf{KIDTR.Enc}(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M)\\ &= (\mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_1}}, M), \cdots, \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_w}}, M)),\end{aligned}$$

where $\mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}$ is partitioned to be w subsets corresponding to nodes $ID_{S_{i_1}}, \cdots, ID_{S_{i_w}}$
In the following games, we will modify step by step the challenge given to the adversary. We define a modified encryption $\mathsf{KIDTR.Enc}^k$ as follow:

$$\begin{aligned}&\mathsf{KIDTR.Enc}^k(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M)\\ &= (\mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_1}}, M_0), \cdots, \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_k}}, M_0),\\ &\quad \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_{k+1}}}, M) \cdots, \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_w}}, M))\end{aligned}$$

Note that

$$\mathsf{KIDTR.Enc}^0(.) = \mathsf{KIDTR.Enc}(.)$$
$$\mathsf{KIDTR.Enc}^k(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M_0) = \mathsf{KIDTR.Enc}(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M_0) \text{ for any } k$$
$$\mathsf{KIDTR.Enc}^w(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M) = \mathsf{KIDTR.Enc}(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M_0) \text{ for any } M$$

Game$_k$ for $k = 1, 2, \ldots, w$: This is the same as in the game $k-1$ with an exception that the challenger use $\mathsf{KIDTR.Enc}^k(.)$ instead of $\mathsf{KIDTR.Enc}^{k-1}(.)$. We call $Adv_{ind,k}^{kidtr-kwibe}(\mathcal{B})$ the advantage of the adversary $\mathcal{B}$ in Game k. We remark that, in the game w, the adversary $\mathcal{B}$ has zero advantage because $\mathcal{B}$ receives two ciphertext of the same message M_0. Therefore, the proof directly holds under the following lemma:

Lemma 22.

$$Adv_{ind,k}^{kidtr-kwibe}(\mathcal{B}) - Adv_{ind,k-1}^{kidtr-kwibe}(\mathcal{B}) \le \epsilon^*,$$

where ϵ^* is the bound on the advantages of the adversaries against KWIBE*.*

Proof. We will construct an adversary $\mathcal{B}'$ that breaks the IND-WID-CPA security of the underlying KWIBE with an advantage of $Adv_{ind,k}^{kidtr-kwibe}(\mathcal{B}) - Adv_{ind,k-1}^{kidtr-kwibe}(\mathcal{B})$.

Setup: The challenger of $\mathcal{B}'$ runs setup algorithm of KWIBE to generate key pair (mpk, msk). It sends mpk to $\mathcal{B}'$ and keeps msk private. $\mathcal{B}'$ passes this mpk to $\mathcal{B}$.

Phase 1: When $\mathcal{B}$ asks three types of key query for a user $id = id_1 \ldots id_l$ in a group ID, $\mathcal{B}'$ sends these queries $WID = (ID, id_1, \ldots, id_l)$ (a $(l+1)-$vector) to its challenger and gets the results. $\mathcal{B}'$ passes the results to $\mathcal{B}$. It assures the correctness because in the construction $d_{ID,id}$ is defined to be d_{WID} in the same way.

For each **Reveal** query on (ID, id), $\mathcal{B}'$ updates the revocation list for group ID by adding id to $\mathcal{R}_{ID}$ (initially empty).

Challenge: The adversary $\mathcal{B}'$ submits two equal length messages M_0, M_1 and an identity ID^*.

The challenger picks a random bit $b \in \{0, 1\}$ and set $C = \text{Encrypt}(msk, ID^*, \mathcal{R}_{ID^*}, M_b)$. The ciphertext C is passed on to the adversary.

$\mathcal{B}'$ partitions $\mathcal{N}_{ID*} \backslash \mathcal{R}_{ID*}$ to $(S_{i_1}, S_{_2}, \cdots, S_{i_w})$ as in the original **Game**$_0$.

$\mathcal{B}'$ submits M_0, M_1 and the identity $ID_{S_{i_k}}$ to its challenger and receives a challenge ciphertext $C_b = \mathsf{KWIBE}.\mathbf{Enc}(\mathsf{msk}, ID_{S_{i_k}}, M_b)$.

$\mathcal{B}'$ encrypts M_0 for identities $ID_{S_{i_1}}, \ldots ID_{S_{i_{k-1}}}$ and encrypts M_1 for identities $ID_{S_{i_{k+1}}}, \ldots ID_{S_{i_w}}$.

$\mathcal{B}'$ finally gives $\mathcal{B}$ the following challenge ciphertext:

$$\begin{aligned}&(\mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_1}}, M_0), \cdots, \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_k}}, M_0),\\ &C_b, \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_{k+1}}}, M) \cdots, \mathsf{KWIBE.Enc}(\mathsf{mpk}, ID_{S_{i_w}}, M))\end{aligned}$$

Phase 2: $\mathcal{B}'$ responses $\mathcal{B}$'s key queries in a similar way to the Phase 1. As $\mathcal{B}$ is not allowed to ask **Leak** query and queries on ID^*, $\mathcal{B}'$ will not make **Leak** query and queries on the targeted identity.

Guess: When $\mathcal{B}$ gives its guess, $\mathcal{B}'$ outputs the same guess. We realizes that, when $b = 0$, the adversary $\mathcal{B}$ exactly plays the **Game**$_{k-1}$ and when $b = 1$, the adversary $\mathcal{B}$ exactly plays the **Game**$_k$. Therefore, the advantage of $\mathcal{B}'$ is $|Adv_{ind,k}^{kidtr-kwibe}(\mathcal{B}) - Adv_{ind,k-1}^{kidtr-kwibe}(\mathcal{B})|$.