Key Leakage Resilience in Trace and Revoke Systems

No Author Given

No Institute Given

Abstract. Trace and revoke systems have been widely studied in theory and implemented in practice (for example, in AACS - a standard for content distribution and digital rights management). 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 at Crypto 2007 and Pirates 2.0 at Eurocrypt 2009, in which pirate decoders could be built from sub-keys of users. Moreover, 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 become very large, possibly beyond the considered threshold in the classical model. In order to countermeasure these types of attacks, it would be desirable to find a way to force traitors to contribute a large portion of their secret information, otherwise their contributions are useless in building pirate decoders. This naturally leads to the consideration of key-leakage resilience in trace and revoke schemes.

In this paper, we present a fully secure key-leakage resilient identity-based trace and revoke system. In order to achieve this goal, we first employ the dual system encryption technique to directly prove the security of a variant of the $\mathsf{BBG-WIBE}$ scheme under known assumptions (and thus avoid a loss of an exponential factor in hierarchical depth in the classical method of reducing the adaptive security of WIBE to the adaptive security of the underlying HIBE). We then modify this scheme to achieve a fully secure key-leakage resilient WIBE scheme for which the security proof makes use of a linear-algebraic technique from [BKKV10] stating that "random subspaces are leakage-resilient". Finally, by using a transformation from a WIBE scheme to a trace and revoke scheme, we propose the first fully secure key-leakage resilient identity-based trace and revoke system. We also notice that the construction of key-leakage resilient WIBE could be useful in its own right.

Keywords: Leakage-resilience, wildcards, revocation, tracing traitors.

1 Introduction

In a system of secure distribution of digital content, a center broadcasts encrypted content to legitimate recipients. Most solutions to this problem can be categorized into broadcast encryption or traitor tracing schemes.

Broadcast encryption systems, independently introduced by Berkovits [Ber91] and Fiat-Naor [FN94], 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 [CFN94], enable the center to trace users who collude to produce pirate decoders.

Trace and Revoke systems [NP00,NNL01] provide the functionalities of both broadcast encryption and traitor tracing.

Identity-based traitor tracing scheme was proposed by Abdalla et al [ADML+07] 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 [PT11] 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 el al [ACD+06] 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 [ADML+07,ZZ11], and under the tree-based framework [PT11]. It is thus important to improve the efficiency and security of the underlying WIBE which is useful in its own right and also has important impact on trace and revoke systems.

1.1 Motivation

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 [KP07] and Pirates 2.0 [BP09], 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. To fight against these types of attack, 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), otherwise the built pirate decoder cannot correctly decrypt. This naturally leads to the consideration of key-leakage resilience in trace and revoke schemes.

Leakage-resilient cryptography has been considered and then formally treated in subsequent works [ISW03,MR04,DLW06,CDD$^+$07,DP07,PSP$^+$08,DP08,SMY09,NS09,KV09,ADW09,BKKV10,LRW11]. 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.

In the context of trace and revoke systems, the key-leakage resilient security model enhances the classical security model. Indeed, the adversary is allowed to make both corruption queries (getting the entire secret keys) and leakage queries to get partial information from any other secret keys. The security should assure that the adversary cannot decrypt any ciphertext intended to any target set that does not contain any corrupted key. In the classical model, the adversary does not have any information about the keys in the target set while in the key-leakage resilient model, the adversary is able to have large amount of information about the keys in the target set.

We note that there are some methods aiming to fight against pirate evolution attacks and pirates 2.0 attack [JL09,DdP11,PT11,ZZ11] but none of these considers a general form of leakage of secret keys. In fact, it is assumed in these methods that the dishonest users leak entire sub-keys which could be used in the encryption procedure.

1.2 Our Contribution

We introduce the first fully secure key-leakage resilient identity-based trace and revoke scheme (that we call a KIDTR scheme). Our KIDTR is a (ℓ_{SK}) - leakage resilient scheme where the leakage in each secret key could be a large fraction of the secret key (precisely, $\frac{\ell_{SK}}{SK} = \frac{(n-1-2c)}{(n+2)(1+c_1+c_3)}$, for any chosen constants c, c_1, c_3 and any $n \geq 3$). Therefore, we could force each dishonest user to contribute a large amount of its secret information to construct a pirate decoder, otherwise its contribution is useless (below the threshold leakage). However, as the secret key of each user is independently chosen from fresh randomness, if a user contributes a large fraction of its secret key, he would lose his anonymity or at least cannot be assured to preserve his anonymity (recall that the anonymity in Pirates 2.0 requires that the contributed information of a user could be also deduced from many other users). We can thus see that key-leakage resilience in trace and revoke systems could be very useful to countermeasure the kind of pirates who collect partial secret informations of users.

Our paper is organized as follow: In Section 2, we first define the framework for key-leakage resilient IDTR system, then we define a new model of adaptive security for a key-leakage resilient IDTR scheme.

In Section 3, we first present the definition of a fully secure key-leakage resilient WIBE scheme, then based on a fully secure key-leakage resilient WIBE scheme, we construct a fully secure key-leakage resilient IDTR scheme.

It turns out that we need to construct a key-leakage resilient WIBE scheme where the leakage is from the secret keys of users (we do not consider the master key's leakage as this is not relevant for trace and revoke systems). Section 4 is devoted to this objective and is the core of our paper.

In Section 4.1, we first construct a $\mathsf{BBG} - \mathsf{WIBE}$ scheme based on composite order groups, and then make use of the dual system encryption technique to directly prove its adaptive security under known assumptions. This avoids a loss of an exponential factor in hierarchical depth when applying the classical method of reducing the adaptive security of WIBE to the adaptive security of the underlying HIBE. This construction could also be seen as a transformation of $\mathsf{BBG}-\mathsf{HIBE}$ in composite order groups [LW10] into $\mathsf{BBG}-\mathsf{WIBE}$ with the same reduction cost in security proofs (both are based on the same assumptions and the reductions are both linear to the total number of adversaries' queries).

Mainly based on this variant of $\mathsf{BBG}-\mathsf{WIBE}$ and inspired by the security proof technique of the key-leakage resilient HIBE in [LRW11], we construct a fully secure key-leakage resilient WIBE scheme in Section 4.2. As WIBE is a generalization of HIBE, the adversary has more freedom in attacking WIBE than in attacking HIBE, especially in two aspects: i) the adversary can choose a challenge pattern (containing wildcards) instead of a challenge identity (which corresponds to a challenge pattern without any wildcard) and ii) the challenge pattern could be matched by many key leakage queries instead of at most one. The first problem could be solved by the classical technique in which the security of WIBE is reduced to the security of HIBE in which the simulator has to correctly guess all the positions of the wildcards in the challenge pattern, thus losing an exponential factor in hierarchical depth. The second problem is rather a new one as our work is the first to consider the key-leakage resilience in WIBE. Fortunately, we show that the dual system encryption technique could be nicely applied to simultaneously solve both problems. Using the dual system encryption technique, step by step, a normal key could be changed to be a semi functional key and if one can manage to deal with a leakage key query matching the challenge pattern, it's not difficult to manage the case where many leakage key queries matching the challenge pattern. Consequently, the security proof of HIBE could be adapted to WIBE with some careful choices of parameters.

2 Key-Leakage Resilient Identity Based Trace and Revoke Systems - KIDTR

2.1 Definition

We follow the same framework of the identity-based traitor tracing (IBTT) in [ADML+07] and the identity-based trace and revoke (IDTR) in [PT11]. 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 $N = 2^l$. 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)**:** The key generation algorithm taking as input security parameter 1^k and number of users for each group N (we assume that the maximum number of users in each group is bounded by N). 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$(\mathsf{mpk}, ID, \mathcal{R}_{ID}, M)$**:** 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 $\mathbf{Dec}(d_{ID,id}, \mathbf{Enc}(\mathsf{mpk}, ID, M)) = M$ with probability one for all $k \in \mathbb{N}, ID, M \in \{0,1\}^*$, $id \in \{0,1\}^l$, $(\mathsf{mpk}, \mathsf{msk}) \overset{\$}{\leftarrow} \mathbf{Setup}(1^k)$ and $d_{ID,id} \overset{\$}{\leftarrow} \mathbf{KeyDer}(\mathsf{msk}, ID, id)$.

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.

2.2 Adaptive Security Model for Key-Leakage Resilient Identity-Based Trace and Revoke Systems.

The adaptive security of KIDTR is defined as follows. In phase 1, the adversary can corrupt secret keys of any user in any group, then these users are revoked from the systems. The adversary can also make leakage query on secret key of any user in any group. In the challenge phase, the adversary chooses a group identity ID^* to attack and it is given a challenge as in a standard IND game. After receiving the challenge, in phase 2, the adversary can corrupt any user of any group except ID^* (otherwise it can decrypt the challenge ciphertext), and it is not allowed to make leakage query on secret keys.

More formally, the key-leakage resilient IND-ID-CPA is defined as follows:

Setup: The challenger takes a parameter k, a maximum number of users in each group N and runs $setup(1^k, N)$ algorithm. The master public key mpk is passed to the adversary. Also, it sets $\mathcal{R}_{ID} = \emptyset$, $\mathcal{T}_{ID} = \{(\langle ID, \rangle, 0, 0)\}$ for all ID. We note that $\mathcal{R}_{ID} \subseteq \langle ID, id\rangle$, and $\mathcal{T}_{ID} \subseteq \langle ID, id\rangle \times \mathcal{SK} \times \mathbb{N}$ (group's identity and user's identity - secret key of the user - leaked bits) for all ID.

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

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 only **Create** and **Reveal**. The adversary is also not allowed to ask for any decryption query (ID, id) in which $ID = ID^*$.

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

Definition 1. *A* KIDTR *scheme is* (ℓ_{SK})*-key leakage secure if all PPT adversaries have at most a negligible advantage in the above security game.*

Note that the above game can extend to handle CCA in the natural way by allowing for decryption queries in phase 1 and phase 2. We call such a game to be the key-leakage resilient IND-ID-CCA game.

3 Generic Construction of Key-Leakage Resilient IDTR

Before presenting the generic construction of KIDTR scheme, we first present the model of adaptive security for a key-leakage resilient WIBE scheme (we call KWIBE).

3.1 Model of Adaptive Security of KWIBE

Our security game in the KWIBE setting follows closely to the MasterLeakHibe game in [LRW11], except we do not consider the case having any leakage in the master key. We let I^* denote the set of all possible identity vectors, $\mathcal{R}$ denote the set of all revealed identities. The new security game, which is called KeyLeakWibe, goes as follows:

Setup: The challenger makes a call to **Setup**(1^λ) and gets the master secret key msk and the public parameters PP. It gives PP to the attacker. Also, it sets $\mathcal{R} = \emptyset$ and $\mathcal{T} = \{(\langle\rangle, 0, 0)\}$. We note that $\mathcal{R} \subseteq I^*$ and $\mathcal{T} \subseteq I^* \times \mathcal{SK} \times \mathbb{N}$ (identity vectors - keys - leaked bits). Thus initially the set $\mathcal{T}$ is empty (empty identity vector and no leakage so far).

Phase 1: In this phase, the adversary can make any of the following queries to the challenger. All of them can be interleaved in any possible way and therefore the input of a query can depend on the outputs of all previous queries (adaptive security).

1. **Create**$(\overrightarrow{I})$: The challenger initially scans $\mathcal{T}$ to find the identity $\overrightarrow{I}$. If this identity exists in $\mathcal{T}$, it responds with $\perp$.
   Otherwise, the challenger makes a call to **Keygen**$(msk\ ,\ \overrightarrow{I}) \to K$ and adds the tuple $(\overrightarrow{I}$, K, 0) to the set $\mathcal{T}$.
2. **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. It is of the form $(\overrightarrow{I}$, SK, $L)$. It checks if $L+ \mid f(\mathrm{SK})| \leq \ell_{SK}$. If this is true, it responds with f(SK) and updates the L in the tuple with $L+| f(\mathrm{SK})|$. If the checks fails, it returns $\perp$ to the adversary.
3. **Reveal**$(\overrightarrow{I})$: Now the adversary requests the entire key with identity $\overrightarrow{I}$. The challenger scans $\mathcal{T}$ to find the requested entry. Let's say the tuple is $(\overrightarrow{I}$, SK, $L)$. The challenger responds with SK and adds the identity vector $\overrightarrow{I}$ to the set $\mathcal{R}$.
4. **Delegate**$(\overrightarrow{I}, I')$: The challenger initially scans $\mathcal{T}$ to find the tuple with identity $\overrightarrow{I}$. Let's say it is $(\overrightarrow{I}, SK, L)$. It makes a call to **Delegate**$(\overrightarrow{I}, SK, I') \leftarrow SK'$ and adds the tuple $(\overrightarrow{I} \mid\mid I', SK', 0)$ to the set $\mathcal{T}$.

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 **Encrypt**$(M_c, \overrightarrow{P^*})$. 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** 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 2. *A* KWIBE *scheme is* (ℓ_{SK})*-key leakage secure if all PPT adversaries have at most a negligible advantage in the above security game.*

3.2 Generic Construction of KIDTR

The construction of KIDTR closely follows the construction of WIBE-IDTR in [PT11], using the new primitive KWIBE instead of WIBE for encryption. We integrate KWIBE to 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$(1^k, N)$**:** Take a security parameter k and the maximum number in each group N (thus $l = \lceil \log_2 N \rceil$). 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 $S_{i_j}, 1 \le j \le w$, 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.

3.3 Security

Theorem 3 (Security of KIDTR).
If the KWIBE *is* (ℓ_{SK}) *- key-leakage secure then our* KIDTR *is also* (ℓ_{SK}) *- key-leakage secure.*

Proof. The proof is provided in Appendix B.

4 Construction of Key-Leakage Resilient WIBE Scheme

4.1 BBG-WIBE Scheme in Composite Order Groups

Among the constructions of WIBE, the construction from Boneh-Boyen-Goh's HIBE scheme is the most efficient (the detail comparison in [ACD+06]). However, this constructions are only efficient in a weak form of security and becomes inefficient when considering adaptive attacks. Recently, Water [Wat09] proposed a dual system encryption technique to improve the security of schemes. There are subsequent works that make use of this technique, one of them proves the Boneh-Boyen-Goh's HIBE scheme fully secure [LW10]. Based on this motivation, we consider a variant of the Boneh-Boyen-Goh's WIBE scheme and using the dual system encryption technique to prove its full security.

When applying the dual system encryption technique to prove the security of the proposed BBG − WIBE scheme in composite order groups, a problem arising is that an attacker can make use of an additional components (the element $C_{3,i}$ added in the transformation from HIBE to WIBE) to distinguish the difference between the distribution of couples (semi-functional key, semi-functional ciphertext) and (normal key, normal ciphertext) or (normal key, semi-functional ciphertext) or (semi-functional key, normal ciphertext) (for examples, the testing $e(K_1, C_{3,i}) == e(E_i, C_2)$ holds in all the cases except (semi-functional key, semi-functional ciphertext)). To overcome this problem, we should manage to impose the distribution of exponents of G_{p_2} components in the semi-functional key and the semi-functional ciphertext compatibly (in above example, making the test alway valid).

We refer the adaptive security model of the Boneh-Boyen-Goh's WIBE scheme in [ACD+06]. The detail construction and the proof of security of the Boneh-Boyen-Goh's WIBE scheme in composite order groups are provided in Appendix C.

4.2 Main Construction From BBG-WIBE

Mainly based on the above variant of BBG − WIBE and inspired from the technique of proving security for a key-leakage resilient HIBE in [LRW11], we now construct a key-leakage resilient WIBE scheme which is (ℓ_{SK})-key leakage secure.
To construct and prove the security, we also choose compatibly the G_{p_2} part of components in the semi-functional key and semi-functional ciphertext as above, this choosing leads to a fully secure

WIBE scheme. On the other hand, to achieve the key-leakage resilient, we make use of the techniques from [BKKV10] and [LRW11] by expanding the space of ciphertext size and key size an additional n dimensional vector, where $n \geq 3$ is a parameter determining the leakage tolerance. Moreover, we also choose compatibly some constants (as r_1, r_2, z_k, z_c) to keep the property that if $\overrightarrow{\Gamma}$ is orthogonal to $\overrightarrow{\delta}$, then the challenge key is nominally semi-functional (and well-distributed as such), and if $\overrightarrow{\Gamma}$ is not orthogonal to $\overrightarrow{\delta}$, then the challenge key is truly semi-functional (and also well-distributed).
The construction is detailled in the following.

Setup $(1^\lambda) \to$ (PP, MK) 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 $\mathbb{Z}_N$. If the maximum depth of the WIBE is D, the setup algorithm chooses a generator $g_1 \overset{\$}{\leftarrow} \mathbb{G}_1$ and a generator $g_3 \overset{\$}{\leftarrow} \mathbb{G}_3$. It picks $b, a_1, \ldots, a_D \overset{\$}{\leftarrow} \mathbb{Z}_N^{D+1}$ 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 $msk = \alpha$, and the public parameters are:

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

Keyder (msk,$(ID_1, ID_2, \ldots, ID_j)$, PP) The key generation 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

$$\begin{aligned} SK &= (\overrightarrow{K_1}, E_{j+1}, \ldots, E_D) = \\ &= \left(\left\langle g_1^{z_1}, g_1^{z_2}, \ldots, g_1^{z_n}, g_1^\alpha \left(h \cdot \prod_{i=1}^j u_i^{ID_i}\right)^{-r} \prod_{i=1}^n g_1^{-x_i z_i}, g_1^r \right\rangle * g_3^{\overrightarrow{\rho}}, \right. \\ &\quad \left. u_{j+1}^r g_3^{\rho_{n+3}}, \ldots, u_D^r g_3^{\rho_{n+2+D-j}} \right) \end{aligned}$$

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

$$\begin{aligned} SK &= (\overrightarrow{K_1}, E_{j+2}, \ldots, E_D) = \\ &= \left(\overrightarrow{K'_1} * \left\langle g_1^{y_1}, g_1^{y_2}, \ldots, g_1^{y_n}, h^{-r'} (E'_{j+1})^{-ID_{j+1}} \left(\prod_{i=1}^{j+1} u_i^{ID_i}\right)^{-r'} \prod_{i=1}^n g_1^{-x_i y_i}, g_1^{r'} \right\rangle * g_3^{\overrightarrow{\rho}'}, \right. \\ &\quad \left. E'_{j+2} u_{j+2}^{r'} g_3^{\rho'_{n+3}}, \ldots, E'_D u_D^{r'} g_3^{\rho'_{n+1+D-j}} \right) \end{aligned}$$

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

$$\begin{aligned} e_{n+2}(\overrightarrow{K_1}, \overrightarrow{C'_1}) = e(g_1, g_1)^{\alpha s} \cdot e(g_1, u_1^{ID_1} \cdots u_j^{ID_j} h)^{-rs} \cdot e(g_1, u_1^{ID_1} \cdots u_j^{ID_j} h)^{rs} \cdot \\ \cdot \prod_{i=1}^n e(g_1, g_1)^{-x_i z_i s} \cdot \prod_{i=1}^n e(g_1, g_1)^{x_i z_i s} = e(g_1, g_1)^{\alpha s} \end{aligned}$$

Notice that in order for the decryption algorithm to work correctly, the identity vector of the secret key has to "*match*" the challenge pattern of the ciphertext. However, if a user's identity is a prefix of this pattern, he can use the delegate algorithm to get the corresponding identity vector as the same length with the pattern and then decrypt correctly.

4.3 Security of Key-Leakage Resilient BBG-WIBE

Semi-functionality

KeygenSF(MK,$(ID_1, ID_2, \ldots, ID_j)) \to \widetilde{K}$. To create a semi-functional key, $\widetilde{K}$, this algorithms calls first **Keygen**(msk,$(ID_1, ID_2, \ldots, ID_j)) \to K$. Then, for K of the form $K = (\overrightarrow{K_1}, E_{j+1}, \ldots, E_D)$, the algorithm picks $z_k \in \mathbb{Z}_N$ and $\overrightarrow{\gamma} = (\gamma_1, \ldots, \gamma_{n+1} = \gamma_{n+2}(z_k - \sum_{i=1}^{j} a_i ID_i), \gamma_{n+2}) \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$. It outputs

$$\widetilde{K} = \left(\overrightarrow{K_1} * g_2^{\overrightarrow{\gamma}}, E_{j+1} g_2^{\gamma_{n+2} a_{j+1}}, \cdots, E_D g_2^{\gamma_{n+2} a_D}\right)$$

EncryptSF(M, $\overrightarrow{P}) \to \widetilde{CT}$. This algorithm first calls the normal encryption algorithm **Encrypt**(M, $\overrightarrow{P}$) to get the ciphertext $CT = (C_0, \overrightarrow{C_1}, C_2)$. Then it picks $z_c \in \mathbb{Z}_N$, and $\overrightarrow{\delta} = (\delta_1, \ldots, \delta_{n+1}, \delta_{n+2} = \delta_{n+1}(z_c + \sum_{i \in \overline{W}(P)} a_i P_i)) \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2}$, 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 $\overrightarrow{\delta^*} = \left(\overrightarrow{\delta} + \left\langle 0, \cdots, 0, \sum_{i \in W(P)} \delta_{n+1} \cdot a_i \cdot ID_i \right\rangle\right)$.

Security properties

In order to prove the security of our system, we first define some similar properties to those in [LRW11]. Namely, the notions of delegate invariance, semi-functional ciphertext invariance, one semi-functional key invariance, and semi- functional security will be adapted to the WIBE setting.
The KeyLeakWibe* game is exactly the same as the KeyLeakWibe game, except that in the KeyLeakWibe* game all the **Delegate** calls are substituted by the **Keygen** calls

Delegation Invariance: We say that a dual system WIBE scheme Π_D has (ℓ_{SK})- *delegate invariance* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the KeyLeakWibe game is negligibly close to the advantage of $\mathcal{A}$ in the KeyLeakWibe* game. We denote this by:

$$\mid Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakWibe}}(\lambda, \ell_{SK}) - Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakWibe}^*}(\lambda, \ell_{SK}) \mid \leq negl(\lambda).$$

We note that any WIBE system where the distribution of delegated keys is identical to the distribution of keys produced by fresh calls to **Keygen** automatically has delegation invariance.

The $\mathsf{KeyLeakC}$ game is exactly the same as the $\mathsf{KeyLeakWibe}^*$ game except that in the **Challenge** phase, the challenger uses **EncryptSF** instead of **Encrypt** to create a semi-functional ciphertext, and returns this to the adversary.

In the $\mathsf{KeyLeakCK}$ game the challenger again uses **EncryptSF** for the challenge phase. And the tuple $\mathcal{T}$ holds for each secret key a normal and a semi-functional version of it. In this game, all keys leaked or given to the attacker are semi-functional.

The $\mathsf{KeyLeak}_b$ game is similar to the $\mathsf{KeyLeakCK}$ game, with the main difference being that the attacker can choose on which version of each key to leak or reveal. In other words, on the first leakage or reveal query on a key of the augmented set $\mathcal{T}$, the attacker tells the challenger whether it chooses the normal or the semi-functional version of the key. In order for the challenger to keep track of the attacker's choice on each key, we further augment each tuple of $\mathcal{T}$ with a lock-value denoted by $\mathcal{V} \in \mathbb{N}$ that can take one of the three values:

– -1: This signifies that the attacker has not made a choice on this key yet and the key is "unlocked". This is the value the tuple gets, in a **Create** query.
– 0: The attacker decides to use the normal version of the key on the first leakage or reveal query on it. All subsequent **Leak** and **Reveal** queries act on the normal version.
– 1: The attacker chooses the semi-functional version and the challenger works as above with the semi- functional version.

To summarize, the tuple is of the form $(X, K, \widetilde{K}, L, V)$ i.e. identity - normal key - semi-functional key - leakage - lock.

At some point, the attacker must decide on a *challenge key* which is "unlocked", V = -1, and tell this to the challenger. The challenger samples a uniformly random bit $b \xleftarrow{\$} \{0, 1\}$ and sets $V = b$. Therefore, the attacker has access to either the normal (if $b = 0$) or the semi-functional (if $b = 1$) version of this key via **Leak** and **Reveal** queries.

The attacker is then allowed to resume queries addressed to either normal or semi-functional keys, with the usual restrictions (i.e. no leakage or reveal queries on keys capable of decrypting the challenge ciphertext after the attacker has seen the challenge ciphertext).

Semi-functional Ciphertext Invariance: We say that a dual system WIBE scheme Π_D has (ℓ_{SK})-*semi-functional ciphertext invariance* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeakWibe}^*$ game is negligibly close to the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeakC}$ game. We denote this by:

$$\mid Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakWibe}^*}(\lambda, \ell_{SK}) - Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakC}}(\lambda, \ell_{SK}) \mid \leq negl(\lambda).$$

Semi-functional Key Invariance: We say that a dual system WIBE scheme Π_D has (ℓ_{SK})-*semi-functional key invariance* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeakC}$ game is negligibly close to the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeakCK}$ game. We denote this by:

$$\mid Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakC}}(\lambda, \ell_{SK}) - Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakCK}}(\lambda, \ell_{SK}) \mid \leq negl(\lambda).$$

One Semi-functional Key Invariance: We say that a dual system WIBE scheme Π_D has (ℓ_{SK})-*one semi-functional key invariance* if, for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeak}_b$ game with $b = 0$ is negligibly close to the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeak}_b$ game with $b = 1$. We denote this by:

$$\mid Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeak}_0}(\lambda, \ell_{SK}) - Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeak}_1}(\lambda, \ell_{SK}) \mid \leq negl(\lambda).$$

Semi-functional Security: We say that a dual system WIBE scheme Π_D has (ℓ_{SK})-*semi-functional security* if for any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the $\mathsf{KeyLeakCK}$ game is negligible. We denote this by:

$$Adv_{\mathcal{A},\Pi_D}^{\mathsf{KeyLeakCK}}(\lambda, \ell_{SK}) \leq negl(\lambda).$$

Theorem 4 (Security of Key-Leakage Resilient $\mathsf{BBG-WIBE}$**).** *Under assumptions* 1, 2, 3 *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 positive constant c is p_2^{-c} is negligible. The length of secret key SK at level i is $(n+2+D-i)(1+c_1+c_3)log(p_2)$ where D is the depth of WIBE, c_1 and c_3 are chosen positive constants in which the security parameter of $\mathbb{G}_1$ and $\mathbb{G}_3$ are $c_1log(p_2)$ and $c_3log(p_2)$. As we can see, the fraction of leakage of secret key at leaf node is the biggest.

Proof. First, we prove that our scheme described above has delegation invariance, semi-functional ciphertext invariance, one semi-functional key invariance, and semi-functional security. We will base each of the properties on the three complexity assumptions 1, 2, and 3.

$\mathsf{KeyLeakWibe} \approx \mathsf{KeyLeakWibe}^*$: 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 secret key **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 secret key **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 above, the nominality of semi components $C_{2,i}$ and semi components E_i is also hidden to the challenger 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.
Similar to the one in [LRW11], 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 $P^* = (P_1, \ldots, P_j)$.
If the challenge key is capable of decrypting the challenge ciphertext 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, 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 [LW10]. One also can see that if the challenge key identity vector *matches* the challenge ciphertext pattern, the secret key is nominal with respect to the ciphertext.

As in [LRW11], based on two following lemma, we get that the change in the simulator's advantage is only negligible.

Lemma 5. *If assumption* 2 *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. As in the proof of lemma 5 in [LW10], if there exists an adversary whose advantage changes by a non-negligible amount under this restriction, 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.

Lemma 6. *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* $\mathcal{A}$*,* $\mathcal{A}$*'s 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 to have an identity vector which matches the challenge ciphertext pattern.*

Proof. As in [LRW11], we suppose 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 [LRW11] with non-negligible advantage (where $m = n + 1$ and $p = p_2$). This will yield a contradiction, since these distributions have a negligible statistical distance.

$\mathcal{B}$ 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 the original master key and generators of all the subgroups, it can make normal as well as semi-functional keys. Hence, it can respond to $\mathcal{A}$'s non-challenge **Phase 1** queries by simply creating the queried keys.

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 (more details on this below). $\mathcal{B}$ 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

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

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 $(ID_1, ID_2, \dots, ID_j)$ and the challenge ciphertext's pattern is $(P_1, P_2, \dots, P_k)$.

$\mathcal{B}$ 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, using
$\overrightarrow{\delta'} = \left(\left\langle \overrightarrow{\delta}, 0 \right\rangle + \langle 0, \dots, 0, 0, \delta_{n+1}(z_c + \sum_{i \in \overline{W}(P)} a_i P_i) \rangle, (\delta_{n+1} a_i)_{i \in W(P)}\right)$ 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 $i = j + 1, \dots, k$, to run the delegation algorithm and get the new $\mathbb{G}_2$ components of the key:

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

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 $\overrightarrow{\varGamma}$ is orthogonal to $\overrightarrow{\delta}$, then the challenge key is nominally semi-functional (and well-distributed as such). If $\overrightarrow{\varGamma}$ 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 it's 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 [LRW11], since these distributions have a negligible statistical distance for f with this output size.
To conclude, the challenger can depends on the difference of advantages in the game $\mathsf{KeyLeak}_0$ and $\mathsf{KeyLeak}_1$ to determine the nature of T.

$\mathsf{KeyLeakC} \approx \mathsf{KeyLeakCK}$: We denote by Q the maximum number of **Create** queries that attacker 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 $\mathsf{KeyLeakC}$ 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 $\mathsf{KeyLeakC}$ game and SF_Q is the $\mathsf{KeyLeakCK}$ 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

$$\mid Adv^{SF_{q^*}}_{\mathcal{A},\Pi_D}(\lambda, \ell_{SK}) - Adv^{SF_{q^*+1}}_{\mathcal{A},\Pi_D}(\lambda, \ell_{SK}) \mid$$

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.

$\mathsf{KeyLeakCK}$ gives no advantage: We use $\mathcal{A}$ with non-negligible advantage in breaking $\mathsf{KeyLeakCK}$ 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 $\mathsf{KeyLeakCK}$ game.

If a dual system encryption WIBE scheme $\Pi_D =$ (**Setup**, **Keygen**, **Encrypt**, **Decrypt**, **KeygenSF**, **EncryptSF**, **Delegate**) has (ℓ_{SK}) - delegation invariance, (ℓ_{SK}) - semi-functional ciphertext invariance, (ℓ_{SK}) - semi-functional key invariance, and (ℓ_{SK}) - semi-functional security, then $\Pi =$ (**Setup**, **Keygen**, **Encrypt**, **Decrypt**) is a (ℓ_{SK}) - key-leakage secure WIBE scheme.
We easily deduce that

$$\mid Adv^{\mathsf{KeyLeakWibe}}_{\mathcal{A},\Pi}(\lambda, \ell_{SK}) - Adv^{\mathsf{KeyLeakCK}}_{\mathcal{A},\Pi_D}(\lambda, \ell_{SK}) \mid$$

is negligible which implies that $Adv^{\mathsf{KeyLeakWibe}}_{\mathcal{A},\Pi}(\lambda, \ell_{SK})$ is negligible. $\square$

References

ACD$^+$06. Michel Abdalla, Dario Catalano, Alex Dent, John Malone-Lee, Gregory Neven, and Nigel Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo 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.

ADML$^+$07. Michel Abdalla, Alexander W. Dent, John Malone-Lee, Gregory Neven, Duong Hieu Phan, and Nigel P. Smart. Identity-based traitor tracing. In Tatsuaki Okamoto and Xiaoyun 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, April 2007.

ADW09. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, August 2009.

Ber91. Shimshon Berkovits. How to broadcast a secret (rump session). In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 535–541. Springer, April 1991.

BP09. Olivier Billet and Duong Hieu Phan. Traitors collaborating in public: Pirates 2.0. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 189–205. Springer, April 2009.

BKKV10. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod 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.

CDD$^+$07. David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 479–498. Springer, February 2007.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, August 1994.

DdP11. P. D'Arco and Angel L. Perez 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.

DLW06. Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 225–244. Springer, March 2006.

DP07. Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *48th Annual Symposium on Foundations of Computer Science*, pages 227–237. IEEE Computer Society Press, October 2007.

DP08. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual Symposium on Foundations of Computer Science*, pages 293–302. IEEE Computer Society Press, October 2008.

FN94. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, August 1994.

ISW03. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, August 2003.

JL09. Hongxia Jin and Jeffrey Lotspiech. Defending against the pirate evolution attack. In *ISPEC '09: Proceedings of the 5th International Conference on Information Security Practice and Experience*, pages 147–158, Berlin, Heidelberg, 2009. Springer-Verlag.

KV09. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, December 2009.

KP07. Aggelos Kiayias and Serdar Pehlivanoglu. Pirate evolution: How to make the most of your traitor keys. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 448–465. Springer, August 2007.

LRW11. Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 70–88. Springer, March 2011.

LW10. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, February 2010.

MR04. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, February 2004.

NNL01. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, August 2001.

NP00. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000: 4th International Conference on Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20. Springer, February 2000.

NS09. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, August 2009.

PSP$^+$08. Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In Masayuki Abe and Virgil Gligor, editors, *ASIACCS 08: 3rd Conference on Computer and Communications Security*, pages 56–65. ACM Press, March 2008.

PT11. Duong Hieu Phan and Viet Cuong Trinh. Identity-based trace and revoke schemes. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec 2011: 5th International Conference on Provable Security*, volume 6980 of *Lecture Notes in Computer Science*, pages 204–221. Springer, October 2011.

SMY09. François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, April 2009.

Wat09. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, August 2009.

ZZ11. Xingwen Zhao and Fangguo 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 [LW10].
Assumption 1 (Subgroup decision problem for 3 primes) Given a group generator $\mathcal{G}$, we define the following distribution:

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

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

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 7. *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 8. *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:

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

$$D = (\mathbb{G}, g, g^{\alpha}X_2, X_3, g^sY_2, Z_2); \quad T_1 = e(g, g)^{\alpha s}, T_2 \xleftarrow{\$} G_T.$$

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 9. *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 Security of the KIDTR Scheme

Our proof follows closely to the proof of theorem 2 in [PT11]. 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 assumption 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 **Creat**, **Leak**, and **Reveal** on identity (ID, id). The challenger responds these queries by running $\mathsf{KIDTR}.\mathbf{Keyder}(\mathsf{msk}, ID, id)$ algorithm to generate the private key $d_{ID,id}$, scanning the tupe $\mathcal{T}$ to compute $f(d_{ID,id})$, and scanning the tupe $\mathcal{T}$ to find $d_{ID,id}$.
$\mathcal{B}$ 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

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

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 10.

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

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

Proof. We will construct an adversary $\mathcal{B}'$ that breaks the $\mathsf{IND\text{-}WID\text{-}CPA}$ 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 $\mathbf{Game}_0$.
$\mathcal{B}'$ 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 $\mathbf{Game}_{k-1}$ and when $b = 1$, the adversary $\mathcal{B}$ exactly plays the $\mathbf{Game}_k$. Therefore, the advantage of $\mathcal{B}'$ is $|Adv^{kidtr-kwibe}_{ind,k}(\mathcal{B}) - Adv^{kidtr-kwibe}_{ind,k-1}(\mathcal{B})|$.

C 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 l denote the maximum depth of the WIBE, the setup algorithm chooses a generator $g \in \mathbb{G}_1$, $X_3 \in \mathbb{G}_3$, $\alpha, b, a_1, \ldots, a_l \xleftarrow{\$} \mathbb{Z}_N$. Denote $u_1 = g^{a_1}, \ldots, u_l = g^{a_l}, h = g^b$. The secret parameter is α, the public parameters are published as:

$$PK = (N, g, h, u_1, \ldots, u_l, X_3, e(g,g)^{\alpha})$$

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

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

Delegate Given a key $K_1', K_2', E_{j+1}', \ldots, E_l'$ 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:

$$\begin{aligned}&K_1 = K_1' g^{r'} \tilde{R}_3, \quad K_2 = K_2'\left(u_1^{ID_1}\cdots u_j^{ID_j} h\right)^{r'} (E_{j+1}')^{ID_{j+1}} u_{j+1}^{r' ID_{j+1}} \tilde{R}_3',\\ &E_{j+2} = E_{j+2}' u_{j+2}^{r'} \tilde{R}_{j+2}, \ldots, E_l = E_l' u_l^{r'} \tilde{R}_l\end{aligned}$$

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:

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

C.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 $(ID_1, \dots, ID_j)$, we first create a normal key $K'_1, K'_2, E'_{j+1}, \dots, E'_l$ using the key generation algorithm. We choose random exponents $\gamma, z_k \xleftarrow{\$} \mathbb{Z}_N$ and output

$$K_1 = K'_1 g_2^{\gamma}, K_2 = K'_2 g_2^{\gamma(z_k + \sum_{i=1}^{j} a_i ID_i)}, E_{j+1} = E'_{j+1} g_2^{\gamma a_{j+1}}, \dots, E_l = E'_l g_2^{\gamma a_l},$$

Semi-functional Ciphertext. A semi-functional ciphertext is created for pattern $(P_1, \dots, P_j)$ 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 $(ID_1, \dots, ID_j)$ matches the pattern $(P_1, \dots, P_j)$ 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 11. *If Assumptions 1, 2, and 3 hold, then our* WIBE *system is 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 [LW10]).

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: In $GameFinal$, β is information-theoretically hidden from the attacker. Hence the attacker can obtain no advantage in breaking the scheme.