This is the full version of the extended abstract which appears in
Proceedings of ProvSec '11 - the Fifth International Conference on Provable Security
(16 October 16 – 18 October, Xi'an, China)
X. Boyen and X. Chen Eds. Springer-Verlag, LNCS 6980, pages 204–221.

Identity-Based Trace and Revoke Schemes

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

[1]LAGA, UMR 7539, CNRS, Université Paris 8, Université Paris 13
[2]Ecole normale supérieure, Paris, France

Abstract. Trace and revoke systems allow for the secure distribution of digital content in such a way that malicious users, who collude to produce pirate decoders, can be traced back and revoked from the system. In this paper, we consider such schemes in the identity-based setting, by extending the model of identity-based traitor tracing scheme by Abdalla et al. to support revocation.
The proposed constructions rely on the subset cover framework. We first propose a generic construction which transforms an identity-based encryption with wildcard (WIBE) of depth $\log(N)$ (N being the number of users) into an identity-based trace and revoke scheme by relying on the complete subtree framework (of depth $\log(N)$). This leads, however, to a scheme with $\log(N)$ private key size (as in a complete subtree scheme). We improve this scheme by introducing generalized WIBE (GWIBE) and propose a second construction based on GWIBE of two levels. The latter scheme provides the nice feature of having constant private key size (3 group elements).
In our schemes, we also deal with advanced attacks in the subset cover framework, namely pirate evolution attacks (PEvoA) and pirates 2.0. The only known strategy to protect schemes in the subset cover framework against pirate evolution attacks was proposed by Jin and Lotspiech but decreases seriously the efficiency of the original schemes: each subset is expanded to many others subsets; the total number of subsets to be used in the encryption could thus be $O(N^{1/b})$ to prevent a traitor from creating more than b generations. Our GWIBE based scheme, resisting PEvoA better than the Jin and Lotspiech's method. Moreover, our method does not need to change the partitioning procedure in the original complete subtree scheme and therefore, the resulted schemes are very competitive compared to the original scheme, with $r\log(N/r)\log N-$size ciphertext and constant size private key.

Keywords: Traitor Tracing, Broadcast Encryption, Subset-cover Framework, Pirate Evolution Attacks, Pirates 2.0

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.

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 and to prevent any set of revoked users from recovering the broadcasted information. Moreover, even if all revoked users collude, they don't 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. The center broadcasts encrypted content so that all users can decrypt this content. The risk here is that different users successfully collude (they are then called traitors) by contributing their private keys to build a pirate decoder and distribute this decoder to illegitimate users. The goal of a traitor tracing system is to allow, when a pirate decoder is found, the authority to run a tracing procedure to identify the traitors.

Trace and Revoke systems [NP00,NNL01] provide the functionalities of both broadcast encryption and traitor tracing. The most famous trace and revoke schemes are those from the subset-cover framework [NNL01], namely complete subtree scheme and subset difference scheme. They have been used as a basis to design the widely spread content protection system for HD-DVDs and Blu-ray disks called AACS [AAC].

Identity-based cryptography was introduced by Shamir in 1984 [Sha85]. However, it's only very recently that identity-based schemes in "one-to-many" settings were introduced, namely identity-based traitor tracing [ADML$^+$07] and identity-based broadcast encryption [Del07,SF07]. In the first identity-based traitor tracing scheme, proposed by Abdalla *et al.* [ADML$^+$07], each group of users has an identity and the encryption is based on Waters' WIBE [ACD$^+$06] and on collusion secure codes [BS95]. The use of collusion secure codes results in unfeasibly large public key and ciphertext sizes. The authors left as an open question the problem of constructing "efficient identity-based traitor tracing scheme, even in the random oracle model". We consider the more general problem of constructing efficient identity-based trace and revoke systems in the standard model.

Recently, two new types of attack have been proposed which point out some weakness of schemes in the subset-cover framework, namely pirate evolution attacks [KP07] and pirates 2.0 [BP09].

Pirate evolution [KP07] is an attack concept against a trace and revoke scheme that exploits the properties of the combined functionality of tracing and revocation in a tree based scheme. Using a set of users' keys, the pirate produces an initial pirate decoder. When this pirate decoder is seized, the pirate evolves the first pirate decoder by issuing a second version that succeeds in decrypting ciphertexts. The same step can be repeated again and again and the pirate continues to evolve a new version of the previous decoder.

Pirates 2.0 attacks [BP09] result from traitors collaborating in a public way. In other words, traitors do not secretly collude but display part of their secret keys in a public place; 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. This type of attacks has been shown to be a real threat for tree based and code based schemes [BP09].

Since schemes from the subset-cover framework have been widely implemented in practice, it's a challenge to build new schemes in this framework that can resist these two type of attacks.

1.1 Our Contribution

We propose the first Identity-based Trace and Revoke system, along with an efficient method for fighting pirates 2.0 and pirate evolution attacks. Our constructions can be considered as variants of the complete subtree scheme. The first scheme makes use of an identity-based encryption with wildcards [ACD+06] (WIBE for short) scheme with linear pirate key size in the depth of the tree. In order to improve this, we first propose a generalized WIBE (GWIBE for short) and then propose a second scheme that relies on 2-level GWIBE. Our second scheme have the following properties:

- It outperforms, in term of efficiency, the identity based traitor tracing scheme in [ADML+07] as it does not need to use a collusion secure code. Note that the scheme in [ADML+07] does not support revocability.
- It resists known pirate evolution attacks [KP07] and pirates 2.0 attacks [BP09].
- It is, asymptotically, very competitive with the original scheme, namely the complete subtree and subset difference schemes (detailed comparisons is given in Table 1.). Moreover, in our scheme, private keys of users are of constant sizes.

1.2 Related Works

Recently, [JL09] proposed a new method to defend against pirate evolution attacks in the subset difference framework. However, their method decreases seriously the efficiency of the original schemes because each subset need to be expanded to many others subsets; the total number of subsets to be used in the encryption becomes sub-linear in the total number of users in the system.

A new method has recently proposed in [DdP11] to defend against pirates 2.0 in the subset-cover framework by integrating the Naor-Pinkas scheme[NP00] into complete subtree scheme (CS) or subset difference scheme (SD). However, because of integrating polynomial interpolation method of Naor-Pinkas [NP00], their method decreases the efficiency of the original schemes and cannot support unbounded number of traitors as in subset-cover schemes.

Finally, in a recently proposed work [ZZ11], the authors also proposed a method to deal with pirates 2.0 in code-based schemes. In fact, their method is essentially similar to the method in [ADML+07] except that they split the second level in [ADML+07] into L levels (L is the length of code). This work shows an interesting result that the scheme [ADML+07] can made to be resistant against pirates 2.0 with almost no additional cost. Note however that code-based schemes actually do not support revocation, they are thus not a trace and revoke system. Table 1 shows that our schemes are by far more efficient than the above mentioned schemes. In fact, our scheme are, asymptotically, very competitive with the original schemes in [NNL01], namely the complete subtree scheme and the subset difference scheme. Because these schemes are widely implemented in practice, our results could also give a significant impact in practice.

We now figure out the general literature about broadcast encryption and traitor tracing. Tree based schemes were independently proposed by Wallner *et al.* [WHA99] and Wong *et al.* [WGL98]. They employ a logical key tree structure and receivers have to refresh their decryption key using information given in broadcasted messages every time users are revoked or a new user joins the system. These are thus *stateful* broadcast schemes and can only be aimed at practical applications where the set of privileged users is rarely updated and in a marginal way.

Stateless broadcast schemes have been introduced in order to avoid the rekeying problem [KRS99], [GSW00,NNL01]. Among these schemes, the NNL schemes [NNL01] seem to be the most practical ones and they were used as a basis to design the widely spread content protection system for HD-DVDs and Blu-ray disks called AACS [AAC].

Some methods for combining key tree structure with other techniques have been proposed: Sherman-McGrew [SM03] used a one-way function, Canetti et al. [CMN99] used a pseudo-random generator, and Kim et al. [KPT00] used a Diffie-Hellman key exchange scheme. Some interesting schemes with constant size private keys were proposed by Asano [Asa02] and [AK05].

Dodis and Fazio [DF02] introduced a way to transform NNL schemes from a secret setting to a public-key setting by using IBE and HIBE. Though based on the same primitive, namely HIBE, each user's private key is still composed by a set of sub-keys and therefore, these schemes suffer from pirates 2.0 and pirate evolution attacks.

In addition to combinatoric schemes, many algebraic schemes with very nice features have been introduced. Kurosawa-Desmedt [KD98] and Boneh-Franklin [BF99] proposed the first elegant algebraic constructions with a deterministic tracing procedure. Boneh and Waters designed a fully collusion resistant trace and revoke scheme [BW06]. This last scheme is very interesting from a theoretical point of view as it is the first scheme that supports full collusion with sub-linear ciphertext size. It is, however, hard to implement in practice because the size of ciphertext is always $O(\sqrt{N})$ whatever the size of collusion, *i.e.* even if there are very few traitors.

Public-key broadcast schemes with constant-size ciphertexts have been proposed in [BGW05] and in [DPP07]. These schemes require decryption keys of size linear in the number of users (a receiver has a constant-size private key, but needs the encryption key to perform a decryption). This storage may be excessive for low-cost devices.

Adaptive security has been investigated in [DF03,GW09]. Attribute based broadcast encryption scheme has recently been proposed in [LS08]

Traitor tracing schemes based on codes have been much investigated since the seminal work of Boneh and Shaw [BS95] on collusion secure codes: Kiayias and Yung [KY02b] proposed a scheme with constant rate. [BP08,BN08] recently achieved constant size ciphertexts. One of the main shortcoming of the code based schemes remains that the user's key is very large (linear in the length of codewords).

1.3 About the Model of PEvoA and Pirates 2.0

We remark that the considerer model of PEvoA and Pirates 2.0 in our paper is not the most general one. However, this model covers the main idea of the attacks presented in [KP07] and [BP09], it also covers the proposed attacks in these original papers. Moreover, the countermeasures mentioned above [JL09], [DdP11] and [ZZ11] are all under our model. Formal definitions of our model of PEvoA and Pirates 2.0 will be given in the paper. Informally speaking, this model requires that the pirate needs to have a sub-key to be able to decrypt a sub-ciphertext. We recall that in combinatoric schemes, a ciphertext is composed by a set of sub-ciphertext each of them is encrypted by a sub-key. The known PEvoA and Pirates 2.0 exploit the fact that a pirate can combine some sub-keys to break the system. The most general model for PEvoA and Pirates 2.0 would be the case that a pirate can combine partial information about sub-keys to break the whole scheme. This model is not considered in this paper neither in previous papers and it is still an open question to find out a countermeasures for the most general form of PEvoA and Pirates 2.0. It appears that this closely relates to the subject of key-leakage resilient cryptography .

2 Fighting Pirates 2.0 and Pirate Evolution Attacks in Tree Based Systems

In this section we present the ideas behind our constructions. The main problem of schemes in the tree-based framework (or more generally, in the combinatoric setting) is that the users' keys contain many sub-keys. Each sub-key corresponds to an internal node of the tree. Consequently, each sub-key can be shared between many users. Thus, if a user only publish sub-keys of a certain level (to ensure that there are many users that share that sub-key), this user cannot be traced but pirates can efficiently collect different sub-keys to build a practical decoder.

Our method for fighting pirates 2.0 and pirate evolution attacks consists of two steps: making private keys indecomposable (the private keys are not composed of sub-keys) and avoiding the derivation of sub-keys of any internal node from the private keys.

2.1 Framework

First Step: Making Private Keys Indecomposable Since the core of pirates 2.0 attacks is the public contribution of partial private key, a natural idea is to render private keys indecomposable. But this condition is not sufficient.

Asano [Asa02] and [AK05] proposed very interesting schemes with constant size private keys. Because private keys in these scheme are of constant size, one might think traitors cannot contribute a partial key information to the public and that pirates 2.0 can then be excluded. However, this is not the case. Indeed, in order for a user U to be able to decrypt a ciphertext at a node in the path from U to the root, U has to be able to derive the corresponding key at that node. This is the reason why these schemes are still vulnerable to pirates 2.0 and pirate evolution attacks. The user U does not need to contribute his key but can compute and contribute any key corresponding to the nodes on the path from U to the root.

Second Step: No Intermediate Key Should be Derived From a Private Key The goal in this step is to construct a scheme that, along with the undecomposability of private keys, enjoying the following additional property: no key at internal nodes can be derived from a private key. This property ensures preventing a traitor to derive and then publish keys at internal nodes and therefore it can only contribute its own key which leaks its identity.

Main Idea: Delegation of Ciphertexts Our main idea is to allow a sub-ciphertext at a node being publicly modifiable to be another sub-ciphertext at any descendant node. This way, by possessing a private key at a leaf, a user can decrypt any sub-ciphertext at its ancestor node. We illustrate the idea in the figure 1. The sub-ciphertext C could be publicly modified to be a ciphertext C_U that can be decryptable by any user U lying in the sub-tree rooted at the corresponding node C. At this stage, we see that the user U could have just one key, this key cannot be used to derive sub-keys for other nodes but it can decrypt ciphertexts at any node in the path from U to the root.

The construction of WIBE in [ACD$^+$06] implicitly gives a delegation of ciphertexts as we need. This is indeed our first solution which is quite inefficient. We then introduce a generalized notion of WIBE and propose a much more efficient solution.

2.2 Solutions

First Solution: Integration of WIBE Into a Complete Subtree Scheme In our framework 2, the first step is to propose a scheme so that a user is unable to derive any key corresponding to its ancestor node. This requirement fits perfectly hierarchical identity based encryption systems (HIBE for short). However, an HIBE system cannot be directly employed for dealing with the second step because a user at a lower level cannot decrypt a ciphertext at a higher level. Fortunately, identity-based

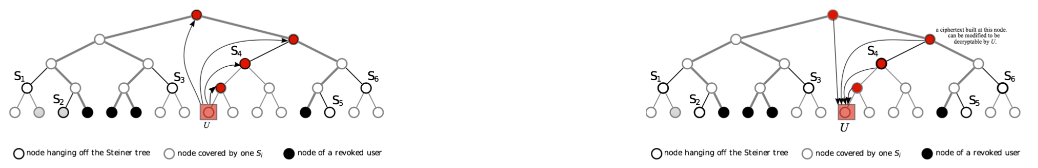
S1
S2
S3
S4
S5
S6
U
a ciphertext built at this node can be modified to be decryptable by U.
node hanging off the Steiner tree
node covered by one S_i
node of a revoked user

Fig. 1. Compairison between Asano's method and our method. In Asano's method, from a private key, a user U can derive keys at any nodes on the path from its leave to the root. Our method can be seen as a dual of Asano's method where a ciphertext at a node can be modified to be decryptable with keys at lower nodes.

encryption with wildcards [ACD$^+$06] (WIBE for short) can be used to solve this issue. The main idea in constructions of current WIBE is to be able to modify the ciphertext so that a legitimate user can put information on the ciphertext to obtain a new ciphertext specifically crafted for him and then can decrypt this modified ciphertext as in HIBE system. This completes the second step. We discuss this with a concrete construction later on.
Our first scheme incorporates a special WIBE with complete subtree structure. In the WIBE used in our first scheme, at each level in hierarchy, there are just two possible identities, say 0 and 1. If we want to encrypt with a node S, we simply put wildcards for lower levels of S and consequent, any user in the subtree rooted by S is able to decrypt. This is illustrated in figure 2.

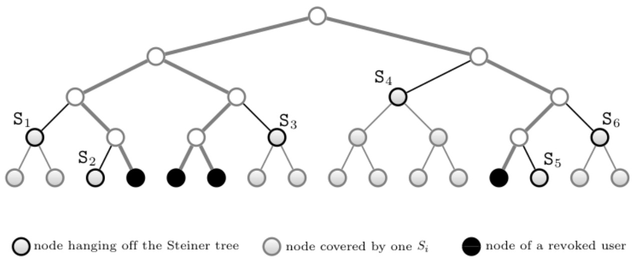

Fig. 2. First scheme based on complete subtree and WIBE: leaves correspond to users, $S_1, \ldots, S_6$ is the covering that excludes revoked users in black while allowing other users to decrypt derived from the steiner tree associated to the set of revoked users R. The WIBE is used to encrypt each identity corresponding to $S_i, i = 1, \ldots, 6$. For S_4 for example, we encrypt with WIBE$(1|0|*|*)$ (the right branch corresponds to bit 1 and the left branch corresponds to bit 0)

The current constructions of WIBE are mainly based on HIBE. Therefore, a private key of a user cannot be used to derive keys of nodes at higher level. This completes the first step. The main idea in constructions of current WIBE is to be able to modify the ciphertext so that a legitimate user can put information on the ciphertext to obtain a new ciphertext specifically crafted for him and then can decrypt this modified ciphertext as in HIBE system. This completes the second step. We discuss this with a concrete construction later on.

Main Solution: Introduction of Generalized WIBE Primitive A shortcoming in our first scheme is that the private key size is linearly in the depth of the tree. To overcome this issue, we propose a new primitive, called generalized WIBE (GWIBE for short) where the wildcard does not need to replace the whole identity at a given level but can be part of the identity at each level. Figure 3 illustrates our method. We group all levels below the root into a single level and put the wildcard as part of the identity. Therefore, we only need a 2-level GWIBE for encryption.

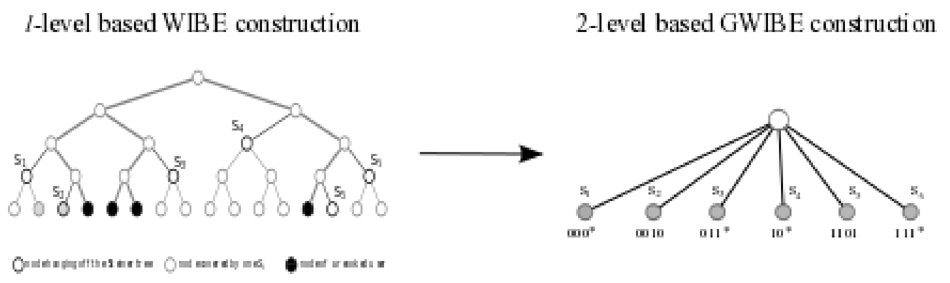
l-level based WIBE construction
2-level based GWIBE construction
S_1
S_2
S_3
S_4
S_5
S_6
000*
0010
011*
10*
1101
111*

Fig. 3. From l-level WIBE based construction to 2-level GWIBE based construction.

2.3 Efficiency Comparison

Besides the property of resisting pirates 2.0 and pirate evolution attacks, our schemes are also very competitive in terms of efficiency. The only previous traitor tracing scheme in the identity based setting is much less efficient than ours, especially when the number of traitors is large.

Compared to the complete subtree and subset difference schemes, the ciphertext in ours schemes is larger (more a factor of $\log(N)$) but our second scheme has constant size private keys.

Dodis and Fazio [DF02] showed a way to transform NNL schemes from a secret setting to a public-key setting by using IBE and HIBE. Their best schemes are denoted by DF-CS (transforming complete subtree scheme to public-key setting by using the Waters IBE in [Wat05]) and DF-SD (transforming subset difference scheme to public-key setting by using the BBG scheme [BBG05]).

A more detailed comparison is given in Table 1.

	Resisting PEvoA /Pirates 2.0	Private key	Public key	Ciphertext	EncTime	DecTime
CS scheme [NNL01]	*no/no*	$O(\log(N))$	0	$O(r\log(\frac{N}{r}))$	$O(r\log(\frac{N}{r}))$	$O(1)$
SD scheme [NNL01]	*no/no*	$O(\log^2(N))$	0	$O(r)$	$O(r)$	$O(1)$
IBTT [ADML+07]	$-/yes$	$O(1)$	$O(c^2\log(N)$ $+\log(1/\epsilon))$	$O(c^2\log(N)$ $+\log(1/\epsilon))$	$O(c^2\log(N)$ $+\log(1/\epsilon))$	$O(\log(N))$
DF-CS (with IBE-Wa)	*no/no*	$O(\log(N))$	$O(\log(N))$	$O(r\log(\frac{N}{r}))$	$O(r\log(\frac{N}{r}))$	O(1)
DF-SD (with BBG)	*no/no*	$O(\log^3(N))$	$O(\log(N))$	$O(r)$	$O(r\log(N))$	$O(\log(N))$
[DdP11] (with CS)	*no/yes*	$O(\log(N))$	O(t)	$O(t\cdot r\log(\frac{N}{r}))$	$O(t\cdot r\log(\frac{N}{r}))$	$O(t)$
[DdP11] (with SD)	*no/yes*	$O(\log^2(N))$	O(t)	$O(t\cdot r)$	$O(t\cdot r)$	$O(t)$
[ZZ11] (with BBG)	$-/yes$	$O(1)$	$O(c^2\log(N)$ $+\log(1/\epsilon))$	$O(c^2\log(N)$ $+\log(1/\epsilon))$	$O(c^2\log(N)$ $+\log(1/\epsilon))$	$O(\log(N))$
Jin-Lotspiech [JL09]	$yes/-$	$O(\log^2(N))$	0	$O(N^{1/b})$	$O(N^{1/b})$	$O(1)$
Wa-WIBE-IDTR 3.6	*yes/yes*	$O(\log(N))$	$O(\log(N))$	$O(r\log(\frac{N}{r})$ $.\log(N))$	$O(r\log(\frac{N}{r})$ $.\log(N))$	$O(\log(N))$
2level-Wa-GWIBE-IDTR 4.4	*yes/yes*	$O(1)$	$O(\log(N))$	$O(r\log(\frac{N}{r})$ $.\log(N))$	$O(r\log(\frac{N}{r})$ $.\log(N))$	$O(\log(N))$

Table 1. Comparison between our schemes and previous schemes. The complete subtree, subset difference schemes, and Jin-Lotspiech method are in the symmetric key setting. The IBTT and [ZZ11] (with BBG) schemes need to fix before set up the maximum number of traitors c and the tracing procedure might have a error with probability ϵ. r denotes the number of revoked users, t denotes the maximum number of colluded users in each subset S_i, '-' denotes "actually not known". Note that the IBTT and [ZZ11] (with BBG) schemes do not support revocation.

3 Construction of Identity-Based Trace and Revoke From WIBE (WIBE-IDTR)

3.1 Background

The definition of an identity-based trace and revoke scheme IDTR, as well as the adaptive security model for IDTR systems, can be found in Appendix A. Our model follows the model of identity-based traitor tracing [ADML+07] in which each group of users has an identity and the broadcaster encrypts messages to a group by using the group's identity. We also deal with revocation, the encryption depends thus on the group's identity and the set of revoked users in that group.We first recall the basic definitions of WIBE, and its security model.

Identity-based encryption with wildcards schemes are essentially a generalization of HIBE schemes where at the time of encryption, the sender can decide to make the ciphertext decryptable by a whole range of users whose identities match a certain pattern. Such a pattern is described by a vector $P = (P_1, ..., P_l) \in (\{0,1\}^* \cup \{*\})^l$, where $*$ is a special wildcard symbol. We say that identity $ID = (ID_1, ..., ID_{l'})$ *matches* P, denoted $ID \in_* P$, if and only if $l^{'} \le l$ and $\forall i = 1...l^{'}$: $ID_i = P_i$ or $P_i = *$. Note that under this definition, any ancestor of a matching identity is also a matching identity. This is reasonable for our purposes because any ancestor can derive the secret key of a matching descendant identity anyway.

More formally, a WIBE scheme is a tuple of algorithms WIBE = (**Setup**, **KeyDer**, **Enc**, **Dec**) providing the following functionality. The **Setup** and **KeyDer** algorithms behave exactly as those of an HIBE scheme. To create a ciphertext of message $m \in \{0,1\}^*$ intended for all identities matching pattern P, the sender computes $C \overset{\$}{\leftarrow} \mathbf{Enc}(\mathsf{mpk}, P, m)$. Any of the intended recipients $ID \in_* P$ can decrypt the ciphertext using its own decryption key as $m \leftarrow \mathbf{Dec}(d_{ID}, C)$. Correctness requires that for all key pairs (mpk, msk) output by **Setup**, all messages $m \in \{0,1\}^*$, all $0 \le l \le L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^l$, and all identities $ID \in_* P$, $\mathbf{Dec}(\mathbf{Keyder}(msk, ID), \mathbf{Enc}(\mathsf{mpk}, P, m)) = m$ with probability one.

Security Model for WIBE. We recall the define of the security of WIBE schemes in [ACD+06]. In the first phase the adversary is run on input the master public key of a freshly generated key pair $(mpk, msk) \overset{\$}{\leftarrow} \mathbf{Setup}(1^k)$. In a chosen-plaintext attack (IND-WID-CPA), the adversary is given access to a key derivation oracle that on input $ID = (ID_1, \cdots, ID_k)$ returns $d_{ID} \overset{\$}{\leftarrow} \mathbf{Keyder}(msk, ID)$.
At the end of the first phase, the adversary outputs two equal length challenge messages m_0, m_1 and a challenge pattern $P^* = (P^*_1, \cdots, P^*_{k^*})$ where $0 \le k^* \le L$, L is the depth of the root tree. The adversary is given a challenge ciphertext $C^* \overset{\$}{\leftarrow} \mathbf{Enc}(mpk, P^*, m_b)$ for a randomly chosen bit b, and is given access to the same oracles as during the first phase of the attack. The second phase ends when the adversary outputs a bit b' . The adversary is said to win the IND-WID-CPA game if $b = b'$ and if it never queried the key derivation oracle for the keys of any identity that matches the target pattern (i.e., any ID such that $ID \in_* P^*$).

Definition 1. A WIBE scheme is (t, q_K, ϵ) IND-WID-CPA-secure if all t-time adversaries making at most q_K queries to the key derivation oracle have at most advantage ϵ in winning the IND-WID-CPA game described above. It is said to be (t, q_K, q_D, ϵ) IND-WID-CCA-secure if all such adversaries that additionally make at most q_D queries to the decryption oracle have advantage at most ϵ in winning the IND-WID-CCA game described above.

3.2 Generic Construction

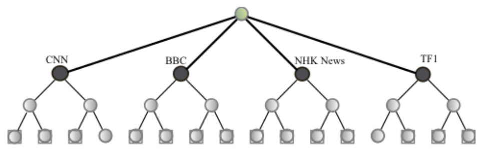

Fig. 4. Identity based Trace and Revoke in Complete Subtree structure

We combine WIBE with the complete subtree method: each group $ID \in \{0,1\}^*$ represents a binary tree and each user $id \in \{0,1\}^l$ (we write $id = id_1 id_2 \cdots id_l$, $id_i \in \{0,1\}$) in a group ID is assigned to be a leaf of the binary tree rooted at ID. For encryption, we will use a WIBE of depth $l+1$, each user is associated with a vector $(ID, id_1, \cdots, id_l)$. This is illustrated in Figure 4.
Concretely, the WIBE-IDTR system works as follows:

Setup$(1^k, N)$**:** Take a security parameter k and the maximum number of user in each group N (thus $l = \lceil \log_2 N \rceil$). Run the setup algorithm of WIBE 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 $E_K : \{0,1\}^* \rightarrow \{0,1\}^*$ and its corresponding decapsulation D_K. The session key K used will be chosen fresh for each message M as a random bit string. E_K should be a fast method and should not expand the plaintext.

Keyder(msk, ID, id)**:** Run the key derivation of WIBE for l+1 level identity $WID = (ID, id_1, id_2, \ldots, id_l)$ (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 an symetric encryption, and outputs as a header the encryption of WIBE for each $ID_{S_{i_1}}, \ldots, ID_{S_{i_w}}$.

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

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 $\mathsf{WIBE}.\mathbf{Enc}(\mathsf{mpk}, ID_{S_{i_j}}, K)$ to obtain K. Finally, compute $D_K(E_K(M))$ to recover the message M.

Trace$^{\mathbb{D}}(\mathsf{msk}, ID)$**:** Our tracing algorithm is similar to the one in [NNL01]. It 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. We note that this is not the strongest notion of traceability. In a black-box traitor tracing, one can identify at least one traitor. Our tracing algorithm relies on the tracing algorithms in NNL framework and target the same notion of traceability. However, unlike the tracing algorithms in NNL, our tracing algorithm can deal with PEvoA and Pirates 2.0, as will be shown in the next sections.

3.3 Security of WIBE-IDTR

Theorem 2. *If the* WIBE *of depth l is* (t', q'_K, ϵ') IND-WID-CPA-*secure, then the* WIBE-IDTR *is* (t^*, q^*_K, ϵ^*) IND-CPA-*secure with*

$$t^* \le t'/(r \log(\frac{N}{r})), \quad q^*_K \le q'_K, \quad \epsilon^* \le r \log(\frac{N}{r}) \times \epsilon'$$

where N is the bound on the number of users and r is the number of revoked users.

The proof is provided in Appendix B. We note that $r \log(\frac{N}{r})$ is the upper-bound on the number of subsets in the partition of non-revoked users. Like in NNL schemes, we should also pay a factor of $r \log(\frac{N}{r})$ in the reduction cost to the security of the chosen primitive for encryption, here is the security of WIBE.

We also present an Instantiation as well as the security analysis of our identity-based trace and revoke from Waters' WIBE in Appendix F. We briefly remark here that, for a general construction of WIBE, the security can be reduced to the security of the underlying primitive HIBE with a lost of an exponential factor (in the hierarchical depth) in the security reduction [ACD+06]. This is due to the fact the simulator has to guess the positions of the wildcards. Fortunately, in our scheme, all the wildcards are regrouped at the end and therefore, the simulator only needs to guess the position of the first wildcard: the lost becomes linear in the hierarchical depth, which is acceptable.

3.4 Resistance to Pirate Evolution Attacks - PEvoA

Pirate evolution is a new kind of attack which decreases the efficiency of the system. In the initial phase, the adversary corrupts t traitor keys and uses this set of keys to create a master pirate box decoder B. B spawns successively a sequence of pirate boxes decoders $B_1, B_2, \ldots$. After a current version of pirate box B_i ($i = 1, \ldots$) is captured and rendered useless, the master box B spawns a new version of pirate box B_{i+1} that can decrypt the ciphertext encrypted at that time. The center continues to run the tracing algorithm to render B_{i+1} useless. The iteration repeats until B cannot spawn a new version of pirate box decoder. If B can spawn a number of pirate boxes greater than t, the system takes a long time to eradicate pirates in this attack. In this case we say that the system is susceptible to the pirate evolution attacks.

The Method of Jin and Lotspiech. Recently, [JL09] proposed a new method to defend against pirate evolution attacks in the subset cover framework but their method seriously decreases the efficiency of the original scheme. We present the detailed analysis in Appendix D.

Formalization of PEvoA. Before proving that our scheme resists to this type of attack, we propose a formalization of pirate evolution attack that covers all known attacks of this type.

Definition 3 (PEvoA in Subset-Cover Systems). In a subset-cover system with N users, each user u_i, $i = 1, \ldots, N$, possesses a set of decryption keys $K_{u_i} = (K_{u_{i,1}}, \ldots, K_{u_{i,h_i}})$. A master pirate box B is built from t traitors $(u_1, \ldots, u_t)$ should possess a set of decryption keys $(K_{u_1}, \ldots, K_{u_t})$.
A pirate evolution attack is an attack in which B spawns successively a sequence of pirate boxes decoders $B_1, B_2, \ldots$ corresponding to each iterations $1, 2, \ldots$. The evolution from the iteration i to the iteration $i+1$ is defined as follows:

- At each iteration i, after the tracing algorithm creates a new partition S_{i+1} which renders the pirate box B_i useless or which revokes some set of users. B spawns a new version B_{i+1} of pirate box, where B_{i+1} must contain at least a decryption key d_{i+1} such that, by using this decryption key d_{i+1}, B_{i+1} can decrypt the ciphertext encrypted based on the new partition S_{i+1} with probability higher than a given threshold q.
- The iteration repeats until B cannot spawn a new version of the pirate box decoder.

The progress of PEvoA can be formalized by an algorithm evo[PEvoA] =PEvoA$(K_{u_1}, \cdots, K_{u_t})$ where $u_1, \cdots, u_t$ are traitors:

```
1. $(K_{u_1}, \cdots, K_{u_N}) \leftarrow \mathbb{G}(1^\lambda, \rho, N)$   where $\rho \leftarrow coins$
2. $K_t \subseteq (K_{u_1}, \cdots, K_{u_N})$   and $|K_t| = t$
3.   $l = 0$
     repeat $l = l+1$
     $B_l(d_{l,1}, \cdots, d_{l,y_l}) \leftarrow B(K_t)$   where $d_{l,1}, \cdots, d_{l,y_l}$ are decryption keys at iteration $l$
     until $Prob[B_l(d_{l,j}, E^{B_1,\cdots,B_{l-1}}(m)) = m] < q$   where $j = 1, \cdots, y_l$
     $E^{B_1,\cdots,B_{l-1}}(m)$ is the ciphertext at iteration $l$, $q$ is given threshold, $m \in \mathbb{M}$
4. output $l$, the number of pirate box decoders that $B$ can spawn.
```

Definition 4 (Resistance to PEvoA). A system is susceptible to PEvoA if the maximum number of pirate box decoders that B can spawn is greater than the number of traitors t. The system is immune from PEvoA if the maximum number of pirate box decoders is less than or equal to t.

In our definition, we closely follow the formalization of [KP07] but we refine the user's private key and evolution step: each user possesses a set of decryption keys, and at each iteration i a pirate box B_i must contain at least a decryption key which is used to decrypt the ciphertext encrypted at that iteration.

This consideration might fail to cover attacks who can derive some partial decryption keys from a set of its key and then could use the partial key to break the security. In any case, our formalization covers the idea of the type of attacks described in [KP07]: each user in [NNL01] possesses a set of decryption keys (a set of long lived keys), and at each iteration master pirate box B spawns a new version of pirate box B_i, where B_i contains at least a decryption key (a long lived key) at some node in the tree.

We also note that the previous schemes in the subset-cover framework are susceptible to pirate evolution attacks because each user can always possess (or derive, in Asano *et. al*'s schemes [Asa02,AK05]) keys corresponding to intermediate nodes in the tree. Therefore, the traitors can easily produce many pirate boxes. This way of attacks are covered in our formalization.

We now show that our WIBE-IDTR scheme resist pirate evolution attacks.

Theorem 5 (Resistance of WIBE-IDTR to Pirate Evolution Attacks). *If there exists a pirate evolution adversary B on* WIBE-IDTR *of depth L, then there exists an adversary B' that breaks the security of* HIBE*.*

Proof. The main idea how our scheme can resist to this type of attacks is come from the property of the underlying HIBE. In our scheme, each user has only a private key corresponding to its identity at depth L. By the property of HIBE, no key at any intermediate node can be derived from the users' key. Therefore, the traitors can only produce pirate box by putting in it their collected keys and the number of pirate box is thus the number of traitors.
Formally, we first prove the following simple lemma.

Lemma 6. *Considering a* WIBE-IDTR *scheme of depth L, if there is an adversary B, by corrupting t user private keys $(d_{ID_1}, \cdots, d_{ID_t})$ for its chosen identities $(ID_1, \cdots, ID_t)$ of depth L, is able to derives a new private key d_{ID^*} for an identity ID^* different than $(ID_1, \cdots, ID_t)$, then one can construct an adversary B' that breaks the semantic security of* HIBE*.*

Proof. The proof is simple, we run B' to simulate B and use the output of B to break the security of HIBE.

- *In the setup phase*, the challenger of B' runs setup algorithm of HIBE to create the public parameters and secret parameters. It gives the public parameters to the B', B' passes these public parameters to the B. The public parameters in two systems are the same.
- *Phase 1:* B chooses arbitrarily t identities $(ID_1, \ldots, ID_t)$ where each identity ID_i $(i = 1, \ldots, t)$ is an identity of depth L. B' queries its challenger key derivation oracle t times on such identities and returns the results to B.
- *The challenge phase:* When B outputs the private key d_{ID^*} corresponding to identity ID^*, B' submits two equal length messages M_0, M_1 and identity ID^* to its challenger. B' can use ID^* as a target identity because ID^* and its prefix never be queried on the phase 1. The challenger picks a random bit $b \in \{0, 1\}$ and set $C = \text{Encrypt}(ID^*, M_b)$. The ciphertext C is passed to B'.
- *Phase 2:* B' uses d_{ID^*} to decrypt C and outputs b.

From the above simulation, we induce that if there exists adversary B then there exists adversary B' with the same probability. □

Assume now that the adversary B corrupts t traitor keys $(d_{u_1}, \ldots, d_{u_t})$. The above lemma shows that B cannot derive a new decryption key $d_{u'}$ not in $(d_{u_1}, \ldots, d_{u_t})$. Therefore, at each iteration i when B spawns a new version of pirate box decoder B_i; B_i must contain at least a new key among the t traitor keys. This implies that the maximum number of pirate box decoders that B can spawn is t. □

3.5 Resistance to Pirates 2.0

Pirates 2.0 is a new type of attack against traitor tracing schemes. This type of attack results from traitors collaborating together *in a public way* and display part of their secret keys in a public place, allowing pirate decoders to be built from this public information. Traitors contribute part of their secret keys in an appropriate way in order to remain anonymous. Moreover, the traitors can contribute information at their discretion and they can publicly collude in groups of very large size.

The Method in [DdP11]. [DdP11] proposed a new method to defend against pirates 2.0 in the subset-cover framework by integrating [NP00] into CS or SD. However, because of using [NP00], their method inherits the weakness of [NP00] scheme, means, only resists maximum t users collude in each subset S_i. We present the detailed analysis in Appendix E.

The Method in [ZZ11]. [ZZ11] also proposed a new method to defend against pirates 2.0 in codebase schemes. However, their method is similar to the method in [ADML$^+$07] except they split the second level in [ADML$^+$07] into L levels (L is the length of code). Note that, their scheme does not support revocability. We present the detailed analysis in Appendix E.

Pirates 2.0. We consider the Pirates 2.0 model for subset-cover framework. Recall that, in this framework, a traitor is a legitimate user u_i who possesses a set of decryption keys $K_{u_i} = (K_{u_{i,1}}, \cdots, K_{u_{i,h_i}})$. We suppose that traitors leaks a subset of their decryption keys to the public domain in a discrete process. When pirates want to decrypt a ciphertext C, they should collect relevant decryption keys from their public environment in order to produce a pirate decoder. We consider that a pirate decoder should contain at least a decryption key that can be based on it to decrypt C. Our consideration, though not in the most general form, exactly reflects the attacks described in [BP09] against subset-cover schemes. We also note that the previous schemes in the subset-cover framework are susceptible to pirates 2.0 attacks because each user can always possess (or derive, in Asano *et. al*'s schemes [Asa02,AK05]) keys corresponding to intermediate nodes in the tree. Therefore, the traitors can easily contribute to the public domain a subset of their decryption keys leading to the production of a pirate decoder. These attack are covered by our considered formalization.

Following the same argument as in the case of pirate evolution attacks, we can show that our WIBE-IDTR scheme resists pirates 2.0.

Theorem 7 (Resistance of WIBE-IDTR to Pirates 2.0). *For any pirates 2.0 adversary B against* WIBE-IDTR *of depth L, there exists an efficient algorithm to output at least a traitor in the collusion that produced B.*

Proof. In our scheme, each user has only a private key corresponding to its identity at depth L. By Lemma 6, no key at any intermediate node can be derived from the users' key. Consider now a traitor u_i contributing a subset of decryption keys S of K_{u_i} to the public domain. In this case, because K_{u_i} is its single decryption key, this immediately implies that S is the leave at user u_i. By definition, the anonymity level of u_i is 1, meaning that u_i is traced to be a traitor.

3.6 Instantiation

We give a concrete construction of IDTR From Waters' WIBE-2 as well as the security analysis of the scheme in Appendix F.

4 Identity-Based Encryption With Generalized Wildcards - GWIBE

Identity-based encryption with generalized wildcards schemes are essentially generalizations of WIBE schemes. The difference is that, in WIBE, at each level, the wildcard is added to replace the whole identity at that level, while in GWIBE, the wildcard might be employed to replace a part of the identity.

More formally, at the time of encryption, the sender can decide to make the ciphertext decryptable by a whole range of users whose identities match a certain pattern. For an l-level GWIBE, such a pattern is described by a vector $P = (P_1, ..., P_l)$. Each P_i, $1 \leq i \leq l$, is a bit string of length n, and can contain a wildcard. The wildcard is determined by its starting position and its final position. For example, for an identity "univ*math" of size 16, the starting position of the wildcard is 5 and the final position is 12, the size of the wildcard is 8. When encrypting with the identity "univ*math", any user who has a key corresponding to identities "univ-paris6.math" or "univ-paris9.math", ... should be able to decrypt.

The starting position is in $\{1, 2, .., n, n+1\}$ with the convention that the value $n+1$ implies no wildcard in P_i. If P_i contains a wildcard (that means a starting position is between 1 and n) we call P_i a generalized wildcard.

At this point, each P_i can be rewritten as $P_i = P_{ip}|| * ||P_{is}$ where P_{ip}, P_{is} are the bit strings of length $ip, is;\ 0 \le ip, is \le n$. The size of the wildcard is $n - ip - is$, its starting position is $ip + 1$ and its final position is $n - is$.

We say that an identity $ID = (ID_1, \cdots, ID_{l'})$ *matches* P, denote $ID \in_* P$, if and only if $l' \le l$ and $\forall i = 1, ..., l' : ID_i = P_i$ or ID_i and P_i have the form $P_i = P_{ip}|| * ||P_{is}$ and $ID_i = P_{ip}||ID_*||P_{is}$ where ID_* is some bit string of length $n - ip - is$.

From the above generalized notions of pattern and matching, the definition and the security model of GWIBE follows exactly the one of WIBE.

The generic construction of IDTR from 2-Level GWIBE is similar to the generic construction of IDTR from WIBE in Section 3.2. The only difference is that, instead of encrypting with WIBE, we will encrypt with 2-level GWIBE. The Theorem 2 can be adapted as follows.

Theorem 8. *If the* GWIBE *of depth* 2 *is* (t', q'_K, ϵ') IND-WID-CPA*-secure, then the 2level-*GWIBE-IDTR *is* (t^*, q^*_K, ϵ^*) IND-CPA*-secure with*

$$t^* \le t'/(r\log(\frac{N}{r})), \quad q^*_K \le q'_K, \quad \epsilon^* \le r\log(\frac{N}{r}) \times \epsilon'$$

where N is the upper bound on the number of users in the system and r is the number of revoked users.

The main contributions in this section are about constructing a GWIBE scheme and then applying it to obtain an efficient trace and revoke system. We now detail these parts.

4.1 Concrete Construction of GWIBE Based on Waters' HIBE (Wa-GWIBE)

We first explain some notation and the high level of the construction. A user's identity is given by a vector $ID = (ID_1, ID_2, \cdots, ID_l)$ where each ID_i is an $n - bit$ string, applying a collision resistant hash function if neccesary. When we write $j \in ID_i$, we mean that the variable j iterates over all bit positions of string of ID_i where the j-th bit of ID_i is one. Using this notation, for $i = 1, \ldots, l$, the Waters' hash function F_i is defined as

$$F_i(ID_i) := u_{i,0} \prod_{j \in ID_i} u_{i,j}$$

where the $u_{i,j}$ are the elements in the master public key.

Now, for a generalized wildcard $ID_{ip}||*||ID_{is}$, we define $F_i(ID_{ip}||*||ID_{is}) := F_i(ID_{ip}||0\ldots0||ID_{is})$ with the number of added bit 0 is the size of the wildcard $*$. The main remark is that if we have the values $u^t_{i,j}$ for all $j = ip + 1, ..., n - is$ and $t \xleftarrow{\$}$, we can compute the values of $F_i(ID_{ip}||ID^*||ID_{is})^t$ from $F_i(ID_{ip}|| * ||ID_{is})^t$ for any bit string ID^* of length $n - ip - is$.

Making use of this property, the main idea is that, if we postpone the computation of elements of wildcard to decryption time by including the separate elements $u^t_{i,j}$ corresponding to wildcards in the ciphertext, each recipient can combine the factors corresponding to his own identity in the decryption.

We extend some notations from WIBE [ACD+06]. Let $P = (P_1, \cdots, P_l)$ is a pattern where P_i is a string of length n, define $W(P)$ be the set containing all generalized wildcard indices in P, i.e. the indices $1 \le i \le l$ such that $P_{ip}|| * ||P_{is}$, $1 \le ip \le n$. Let $\overline{W}(P)$ be the complementary set containing all non-generalized wildcard indices in P. Clearly $W(P) \cap \overline{W}(P)) = \phi$ and $W(P) \cup \overline{W}(P) = \{1, ..., l\}$.

Let $\mathbb{G}, \mathbb{G}_T$ be multiplicative groups of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \leftarrow \mathbb{G}_T$.

Setup: The trusted authority chooses randomly generators $g_1, g_2, u_{1,0}, \cdots, u_{L,n} \xleftarrow{\$} \mathbb{G}^*$, and a random value $\alpha \xleftarrow{\$} \mathbb{Z}_p$, where L is the maximum hierachy depth. Next, it computes $h_1 \leftarrow g_1^\alpha$ and $h_2 \leftarrow g_2^\alpha$. The master public key is $mpk = (g_1,\ g_2,\ h_1,\ u_{1,0}, \cdots, u_{L,n})$, and the master secret key is $msk = h_2$. For each level $i = 1, \ldots, L$, a function $F_i : \{0, 1\}^n \to \mathbb{G}^*$ is defined as:

$$F_i(ID_i) = u_{i,0} \prod_{j \in ID_i} u_{i,j}$$

Keyder: An identity of a user is given by a vector $ID = (ID_1, \cdots, ID_l)$, where ID_i is a bit string of length n (applying a collision-resistant hash function if necessary). First, random values $r_1, \cdots, r_l \overset{\$}{\leftarrow} \mathbb{Z}_p$ are chosen, then the private key d_{ID} is computed as

$$(a_0, a_1, \cdots, a_l) = \left(h_2 \prod_{i=1}^{l} F_i(ID_i)^{r_i}, g_1^{r_1}, \cdots, g_1^{r_l}\right)$$

A secret key for identity $ID = (ID_1, ID_2, \cdots, ID_l)$ can be computed by its parent with identity $ID = (ID_1, ID_2, \cdots, ID_{l-1})$ as follows. Let $(a_0, a_1, \cdots, a_{l-1})$ be the parent's secret key. It chooses $r_l \overset{\$}{\leftarrow} \mathbb{Z}_p$ and outputs

$$d_{ID} = (a_0 \times F_l(ID_l)^{r_l}, a_1, \cdots, a_{l-1}, g_1^{r_l})$$

Enc: To encrypt a message $m \to \mathbb{G}_T$ to all identities matching pattern $P = (P_1, \cdots, P_l)$, the sender chooses $t \overset{\$}{\leftarrow} \mathbb{Z}_p$ and outputs the ciphertext $C = (P, C_1, C_2, C_3, C_4)$, where

$$C_1 \leftarrow g_1^t \qquad C_2 \leftarrow (C_{2,i} = F_i(P_i)^t)_{i \in \overline{W}(P)} \qquad C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^t$$

$$C_4 \leftarrow (C_{4,ip,is} = F_i(P_{ip}||0...0||P_{is})^t, (C_{4,i,j} = u_{i,j}^t)_{j=ip+1,\ldots,n-is})_{i \in W(P)}$$

Where $P_{ip}||0...0||P_{is}$ is a string of length n, or the number of bit 0 is $n - ip - is$. That means we replace the elements of wildcard in identity P_i by bit 0.

Dec: If the receiver is the root authority holding the master key $msk = h_2$, then it can recover the message by computing $C_3/\hat{e}(C_1, h_2)$. Any other receiver with identity $ID = (ID_1, \cdots, ID_l)$ matching the pattern P to which the ciphertext was created (i.e., $ID \in_* P$) can decrypt the ciphertext $C = (P, C_1, C_2, C_3, C_4)$ by computing
$C_2' = (C_{2,i}')_{i=1,\ldots,l}$ as

$$C_{2,i}' = F_i(ID_i)^t \leftarrow \begin{cases} C_{2,i} & \text{if } i \in \overline{W}(ID) \\ C_{4,ip,is} \times \prod_{j=ip+1,\ldots,n-is}^{j \in ID_i} C_{4,i,j} & \text{if } i \in W(ID) \end{cases}$$

then computes

$$C_3 \cdot \frac{\prod_{i=1}^{l} \hat{e}(a_i, C_{2,i}')}{\hat{e}(C_1, a_0)} = m \cdot \hat{e}(h_1, g_2)^t \cdot \frac{\prod_{i=1}^{l} \hat{e}(g_1^{r_i}, F_i(ID_i)^t)}{\hat{e}(g_1^t, h_2 \cdot \prod_{i=1}^{l} F_i(ID_i)^{r_i})}$$

$$= m \cdot \hat{e}(h_1, g_2)^t \cdot \frac{\prod_{i=1}^{l} \hat{e}(g_1^{r_i}, F_i(ID_i)^t)}{\hat{e}(g_1^t, h_2) \cdot \hat{e}(g_1^t, \prod_{i=1}^{l} F_i(ID_i)^{r_i})}$$

$$= m \cdot \frac{\hat{e}(g_1^{\alpha}, g_2)^t}{\hat{e}(g_1^t, g_2)^{\alpha}} \cdot \frac{\prod_{i=1}^{l} \hat{e}(g_1^{r_i}, F_i(ID_i)^t)}{\prod_{i=1}^{l} \hat{e}(g_1^t, F_i(ID_i)^{r_i})} = m$$

4.2 Security Analysis

The proof of the following theorem can be found in Appendix C.

Theorem 9. *If the* Wa-HIBE *of depth L is (t, q_K, ϵ)* IND-ID-CPA*-secure, then the* Wa-GWIBE *scheme of depth L is (t', q_K', ϵ')* IND-GWID-CPA*-secure*

$$t' \leq t - L \cdot n(1 + q_K) \times t_{exp}, \quad q_K' \leq q_K, \quad \epsilon' \leq \epsilon \times n^{2 \times L}$$

where t_{exp} is the time taken to perform an exponentiation in $\mathbb{G}$.

The Wa-GWIBE *used in our Trace and Revoke system.* We first remark that the above reduction of a general construction of GWIBE is extremely costly. Fortunately, in our construction of a trace and revoke system, we only need to a 2-level Wa-GWIBE and we can get a much more efficient reduction. Let us briefly explain the presence of the factor $n^{2\times L}$ in the reduction of the above theorem. In fact, at each level the simulator has to guess the start position and the final position of wildcard (note that if the start position is $n+1$ then it means there is no wildcard at this level). The probability of a good guess at a level is $1/n^2$. Because there are L levels, therefore the probability that A guesses $W(P^*)$ correctly is $1/n^{2\times L}$. We now see how we can reduce this reduction cost for the purpose of a trace and revoke system. Indeed, we only need to a 2-level Wa-GWIBE and moreover, the first layer (the group identity) does not contain wildcard. The used pattern P used in encryption always has the form $P=(P_1, P_{2p}||*)$. Therefore, in the proof of the above theorem, the simulator does not need to guess the position of wildcard in the first layer, and at the second level, $\mathcal{B}$ needs only to guess the starting position of the wildcard. Therefore, the factor $1/n^{2\times L}$ becomes $1/n$ in this case. This is acceptable because $n=\log(N)$ where N is the number of users in the system. We conclude that the (t', q'_K, ϵ') IND-GWID-CPA security of 2-level Wa-GWIBE can thus be reduced to a (t, q_K, ϵ) IND-ID-CPA-security of a 2-level Wa-HIBE with:

$$t' \leq t - \log(N)(1+q_K)\times t_{exp}, \quad q'_K \leq q_K, \quad \epsilon' \leq \epsilon \times \log(N). \tag{1}$$

4.3 Generic Construction of IDTR From 2-Level GWIBE (2level-GWIBE-IDTR)

Setup(1^k)**:** Take a security parameter k and the maximum number of user in each group N. Run the setup algorithm of GWIBE with the security parameter k and the hierarchical depth $l=2$ which returns (mpk, msk). The setup then outputs (mpk, msk).
Similar to the scheme based on WIBE 3.2, the setup also defines a method (E_K, D_K) to encrypt the message itself.

Keyder(msk, ID, id)**:** Run the key derivation of GWIBE for 2 level identity $WID=(ID, id)$ and get the decryption key d_{WID}. Output $d_{ID,id}=d_{WID}$.

Enc(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 a 2 dimension vector identity $ID_{S_{i_j}}=(ID, id_{i_j}||*)$ where $id_{i_j}=id_{i_j,1},\ldots,id_{i_j,k}$ is the path from the root ID to the node S_{i_j} and the size of the wildcards $*$ is $n-k$.
The encryption algorithm randomly chooses a session key K, encrypts M under the key K by using an symetric encryption, and outputs as a header the encryption of 2-Level GWIBE for each $ID_{S_{i_1}},\ldots,ID_{S_{i_w}}$:

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

Dec$(d_{ID,id}, C)$**:** A user receives 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 GWIBE.**Enc**(mpk, $ID_{S_{i_j}}, K$) to obtain K. Finally, compute $D_K(E_K(M))$ to recover the message M.

Trace$^{\mathbb{D}}$(msk, ID)**:** Our tracing algorithm is similar to the one in [NNL01]. Take as input msk, ID, an illegal decryption box $\mathbb{D}$, return either a subset consisting the traitors or a new partition of $\mathcal{N}_{ID}/\mathcal{R}_{ID}$ that renders the illegal decryption box useless.

Theorem 10. *If the* GWIBE *of depth* 2 *is* (t', q'_K, ϵ') IND-WID-CPA-*secure, then the 2level-*GWIBE-IDTR *is* (t^*, q^*_K, ϵ^*) IND-CPA-*secure with*

$$t^* \leq t'/(r\log(\frac{N}{r})), \quad q^*_K \leq q'_K, \quad \epsilon^* \leq r\log(\frac{N}{r})\times\epsilon'$$

where N *is the upper bound on the number of users in the system and* r *is the number of revoked users.*

Proof. We deduce the proof of this theorem from the proof of Theorem 2.

4.4 Construction of IDTR from 2-Level Wa-GWIBE (2level-Wa-GWIBE-IDTR)

Applying the above construction of 2-level-Wa-GWIBE to the generic construction of IDTR from 2-level-GWIBE, we obtain a 2level-Wa-GWIBE-IDTR. We give the construction in Appendix G.

Concerning the security of the scheme, from Theorem 9 and the equation 1 which precisely states the security of the 2-level Wa-GWIBE used in our construction of trace and revoke system, and from Theorem 10, we directly obtain the following result:

Theorem 11. *If Waters' HIBE of depth 2 is (t, q_K, ϵ) IND-WID-CPA-secure, then the 2level-Wa-GWIBE-IDTR is (t^*, q_K^*, ϵ^*) IND-CPA-secure with*

$$t^* \leq (t - \log(N)(1 + q_K)/(r \log(\frac{N}{r})), \quad q_K^* \leq q_K, \quad \epsilon^* \leq r \log(\frac{N}{r}) \times \epsilon \times \log(N)$$

In addition, it is proved in [ACD⁺06] that: if there exists a (t, q_K, ϵ) adversary against Waters' HIBE of depth L, then there exists an algorithm solving the BDDH problem with advantage $\epsilon_{\mathsf{BDDH}} = O(\epsilon/(nq_K)^L)$. In our scheme, we only use 2-level Waters' HIBE, and since $n \leq \log(N)$, we have $\epsilon_{\mathsf{BDDH}} = O(\epsilon/\log^2(N)q_K^2)$.

In conclusion, the security of our scheme can be based on the hardness of the BDDH problem:

Theorem 12. *If the BDDH is $(\epsilon_{\mathsf{BDDH}})$ secure, then the 2level-Wa-GWIBE-IDTR is (q_K^*, ϵ^*) IND-CPA-secure with*

$$\epsilon^* = O(r \log(\frac{N}{r}) \times \epsilon_{\mathsf{BDDH}} \times (q_K^*)^2 \times \log^3(N)).$$

4.5 Resistance to PEvoA and Pirates 2.0

Under similar arguments for the WIBE-IDTR scheme, our construction of 2level-GWIBE-IDTR resists to PEvoA and pirates 2.0.

Again, the main reason comes from the same proof that the adversary cannot derive a new decryption key at any intermediate node from a set of traitors' secret keys, otherwise the security of the underlying scheme (here is 2level-GWIBE) could be broken, as shown in lemma 6.

Acknowledgments

This work was supported in part by the French ANR-09-VERS-016 BEST Project.

References

AAC. AACS LA. AACS Specifications. At `http://www.aacsla.com/specifications/`.

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, Part II*, volume 4052 of *LNCS*, 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*, volume 4450 of *LNCS*, pages 361–376. Springer, April 2007.

Asa02. Tomoyuki Asano. A revocation scheme with minimal storage at receivers. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 433–450. Springer, December 2002.

AK05. Tomoyuki Asano and Kazuya Kamio. A tree based one-key broadcast encryption scheme with low computational overhead. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 05*, volume 3574 of *LNCS*, pages 89–100. Springer, July 2005.

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

BP08. Olivier Billet and Duong Hieu Phan. Efficient Traitor Tracing from Collusion Secure Codes. In Reihaneh Safavi-Naini, editor, *Information Theoretic Security—ICITS 2008*, volume 5155 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2008.

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

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, May 2005.

BF99. Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, August 1999.

BGW05. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, August 2005.

BN08. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08*, pages 501–510. ACM Press, October 2008.

BS95. Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer, August 1995.

BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 211–220. ACM Press, October / November 2006.

CMN99. Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 459–474. Springer, May 1999.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, 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.

Del07. Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 200–215. Springer, December 2007.

DPP07. Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *PAIRING 2007*, volume 4575 of *LNCS*, pages 39–59. Springer, July 2007.

DF02. Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *In Digital Rights Management 02, volume 2696 of LNCS*, pages 61–80. Springer, 2002.

DF03. Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 100–115. Springer, January 2003.

FN94. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, August 1994.

GSW00. Juan A. Garay, Jessica Staddon, and Avishai Wool. Long-lived broadcast encryption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 333–352. Springer, August 2000.

GW09. Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, April 2009.

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.

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

KY02a. Aggelos Kiayias and Moti Yung. On Crafty Pirates and Foxy Tracers. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management—DRM 2001*, volume 2320 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 2002.

KY02b. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, April / May 2002.

KPT00. Yongdae Kim, Adrian Perrig, and Gene Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In S. Jajodia and P. Samarati, editors, *ACM CCS 00*, pages 235–244. ACM Press, November 2000.

KRS99. Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 609–623. Springer, August 1999.

KD98. Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 145–157. Springer, May / June 1998.

LS08. David Lubicz and Thomas Sirvent. Attribute-based broadcast encryption scheme made efficient. In Serge Vaudenay, editor, *AFRICACRYPT 08*, volume 5023 of *LNCS*, pages 325–342. Springer, June 2008.

NNL01. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, August 2001.

NP00. Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, February 2000.

SF07. Ryuichi Sakai and Jun Furukawa. Identity-based broadcast encryption. Cryptology ePrint Archive, Report 2007/217, 2007. `http://eprint.iacr.org//217`.

Sha85. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, August 1985.

SM03. Alan T. Sherman and David A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Softw. Eng.*, 29(5):444–458, 2003.

WHA99. D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. In *RFC 2627*, 1999.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.

WGL98. Chung Kei Wong, Mohamed G. Gouda, and Simon S. Lam. Secure group communications using key graphs. In *Proceedings of ACM SIGCOMM*, pages 68–79, Vancouver, BC, Canada, August 31 – September 4, 1998.

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 Identity Based Trace and Revoke Systems

A.1 Framework

We follow the framework of the identity-based traitor tracing in [ADML$^+$07]: broadcast groups are referred to by an identity string $ID \in \{0,1\}^*$, individual users are referred to by an identity $id \in \{0,1\}^l$. The maximum number in each group $N = 2^l$ is fixed like many previous schemes (we can choose, for example, $l = 32$ and the potential number in each group can reach four billions, that is sufficient in practice). To make user i member of the group ID, the trusted key distribution centre provides it with a personal decryption key $d_{ID,id}$. In order to support revocation, let $\mathcal{N}_{ID}$ be the set of all users in the group ID and $\mathcal{R}_{ID}$ be a group of users whose decryption privileges should be revoked. The goal of our system is to allow anyone to encrypt a message to the general group ID such that any user $u \in \mathcal{N}_{ID} \backslash \mathcal{R}_{ID}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}_{ID}$ cannot decrypt it.

Formally, a 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.
We note that this is not the strongest notion of traceability. In a black-box traitor tracing, one can identify at least one traitor. Our tracing algorithm relies on the tracing algorithms in NNL framework and target the same notion of traceability. Concerning the very weak notion of white-box tracing, we remark that the previous tree-based schemes do not fully achieve this level since the traitors could publish sub-keys at internal nodes. In our scheme, we will show that the only way for users to decrypt ciphertexts is to use their keys at leaves (private keys) and this directly reveal the identities of traitors, in the white-box tracing.

We shall assume that all "pirate" decryption boxes are resettable [KY02a], meaning that they retain no state between decryptions. In particular, pirate boxes cannot self-destruct.

A.2 Adaptive Security Model for Identity-Based Trace and Revoke Systems.

The adaptive security of IDTR is defined as follows. First, the adversary can corrupt secret keys of any user in any group. These users are revoked from the systems. Then the adversary can choose a group identity ID to attack and it is given a challenge as in a standard IND game. After receiving the challenge, the adversary can corrupt any user of any group but ID^* (otherwise it can decrypt the challenge ciphertext).

More formally, the $\mathsf{IND\text{-}ID\text{-}CPA}$ 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.

Phase 1: The adversary access to a key extraction oracle which it can query on arbitrary pairs (ID, id) of group identities ID and user identity id, in an adaptive way, and gets the decryption keys $d_{ID,id}$. For each query on (ID, id), the challenger update the revocation list for group ID by adding id to $\mathcal{R}_{ID}$ (initially empty).

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 adversary is not allowed to ask for any decryption query (ID, id) in which $ID = ID^*$.

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

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 IND-ID-CCA game.

The advantage $\mathbf{Adv}^{\text{ind-id-cpa}}_{\mathcal{A},\mathsf{IDTR}}(k)$, respectively $\mathbf{Adv}^{\text{ind-id-cca}}_{\mathcal{A},\mathsf{IDTR}}(k)$, of an adversary $\mathcal{A}$ in breaking the indistinguishability of scheme IDTR is defined as the probability of $\mathcal{A}$ winning the corresponding game minus one-half. We say that the traitor tracing scheme is IND-ID-CPA, respectively IND-ID-CCA secure, if this advantage is a negligible function in k for any adversary $\mathcal{A}$ with running time polynomial in k.

A.3 Subset Cover Framework [NNL01]

Let $\mathcal{N}$ be the set of all users, $|\mathcal{N}| = N$, and $\mathcal{R} \in \mathcal{N}$ be a group of $|\mathcal{R}| = \text{r}$ users whose decryption privileges should be revoked. The goal of a revocation algorithm is to allow a user to transmit a message M to all users such that any user $u \in \mathcal{N}/\mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ cannot decrypt it.

A system consists of three parts: (1) An initiation scheme, which is a method for assigning the receivers secret information that will allow them to decrypt. (2) The broadcast algorithm - given a message M and the set $\mathcal{R}$ of users that should be revoked outputs a ciphertext message $M^{'}$ that is broadcast to all receivers. (3) A decryption algorithm - a (non-revoked) user that receives ciphertext $M^{'}$ using its secret information should produce the original message M. Since the receivers are stateless, the output of the decryption should be based on the current message and the secret information only.

In this framework an algorithm defines a collection of subsets $(S_1, S_2, \cdots, S_w) \subset \mathcal{N}$. Each subset S_i, $1 \le i \le w$, has an identity. Each member u of S_i should be able to decrypt the ciphertext which is encrypted by identity of S_i. Given a revoked set $\mathcal{R}$, the remaining users $\mathcal{N}/\mathcal{R}$ are partitioned into disjoint subsets $(S_{i_1}, S_{i_2}, \cdots, S_{i_w})$, and $S_{i_1} \cup S_{i_2} \cup, \cdots, \cup S_{i_w} = \mathcal{N}/\mathcal{R}$

And a session key K is encrypted w times with identities of w subsets $(S_{i_1}, S_{i_2}, \cdots, S_{i_w})$.

Specifically, an algorithm in the framework uses two encryption schemes:

- A method E_K: $\{0, 1\}^* \to \{0, 1\}^*$ to encrypt the message itself. The key K used will be chosen fresh for each message M - a session key - as a random bit string. E_K should be a fast method and should not expand the plaintext. The simplest implementation is to Xor the message M with a stream cipher generated by K.
- A method $Enc(S_{i_j})$ to encrypt the session key K for the receivers in the subset S_{i_j}.

A.4 Generalized WIBE

Very similar to a WIBE, a GWIBE scheme is a tuple of algorithms GWIBE = (**Setup**, **KeyDer**, **Enc**, **Dec**) providing the following functionality. The **Setup** and **KeyDer** algorithms behave exactly as those of an HIBE scheme. To create a ciphertext of message $m \in \{0, 1\}^*$ intended for all identities matching pattern P, the sender computes $C \overset{\$}{\leftarrow} \mathbf{Enc}(\mathsf{mpk}, P, m)$. Any of the intended recipients $ID \in_* P$ can decrypt the ciphertext using its own decryption key as $m \leftarrow \mathbf{Dec}(d_{ID}, C)$. The correctness requires that for all key pairs (mpk, msk) output by **Setup**, all messages $m \in \{0, 1\}^*$, all $0 \le l \le L$, all patterns P as above, and all identities $ID \in_* P$, $\mathbf{Dec}(\mathbf{Keyder}(msk, ID), \mathbf{Enc}(\mathsf{mpk}, P, m)) = m$ with probability one.

Security Model for GWIBE. In the first phase the adversary is run on input the master public key

of a freshly generated key pair $(mpk, msk) \xleftarrow{\$} \textbf{Setup}$. In a chosen-plaintext attack (IND-GWID-CPA), the adversary is given access to a key derivation oracle that on input $ID = (ID_1, \cdots, ID_k)$ returns $d_{ID} \xleftarrow{\$} \textbf{Keyder}(msk, ID)$.
At the end of the first phase, the adversary outputs two equal length challenge messages m_0, m_1 and a challenge pattern $P^* = (P^*_1, \cdots, P^*_{k^*})$ where $0 \le k^* \le L$, L is the depth of the root tree. The adversary is given a challenge ciphertext $C^* \xleftarrow{\$} \textbf{Enc}(mpk, P^*, m_b)$ for a randomly chosen bit b, and is given access to the same oracles as during the first phase of the attack. The second phase ends when the adversary outputs a bit b' . The adversary is said to win the IND-GWID-CPA game if $b = b'$ and if it never queried the key derivation oracle for the keys of any identity that matches the target pattern (i.e., any ID such that $ID \in_* P^*$).

Definition 13. A GWIBE scheme is (t, q_K, ϵ) IND-GWID-CPA-secure if all t-time adversaries making at most q_K queries to the key derivation oracle have at most advantage ϵ in winning the IND-GWID-CPA game described above.

B Proof of Theorem 2

We organize our proof as a sequence of games. The first game **Game 0** defined will be the real identity-based trace and revoke 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 WIBE.

Game 0: This is the real attack game of an adversary $\mathcal{B}$ against the proposed identity based trace and revoke system. After receiving the public key mpk, $\mathcal{B}$ issues adaptively key derivation queries (ID, id). The challenger responds by running IDTR.**Keyder**(msk, ID, id) algorithm to generate the private key $d_{ID,id}$ and passes $d_{ID,id}$ to $\mathcal{B}$.
$\mathcal{B}$ 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 by the adversary $\mathcal{B}$.
The challenger picks then a random bit $b \in \{0, 1\}$ and set $C = \mathsf{IDTR.Enc}(msk, \mathcal{N}_{ID^*} \backslash \mathcal{R}_{ID^*}, M_b)$. 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 $b' = b$. More precisely, $\mathcal{B}$ wins if its advantages, defined as below, of guessing the bit b is non-negligeble

$$Adv^{idtr-wibe}_{ind}(t) = |2 \times Pr[\mathcal{B}(\mathsf{IDTR.Enc}(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M_b)) = b] - 1|$$

which can be equivalently written as:

$$\begin{aligned} & Adv^{idtr-wibe}_{ind}(\mathcal{B}) \\ & = |Pr[\mathcal{B}(\mathsf{IDTR.Enc}(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M_1)) = 1] - Pr[\mathcal{B}(\mathsf{IDTR.Enc}(msk, \mathcal{N}_{ID^*}/\mathcal{R}_{ID^*}, M_0)) = 1]| \end{aligned}$$

In our construction, the encryption of trace and revoke system is performed as:

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

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 $\mathsf{IDTR.Enc}^k$ as follow:

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

Note that

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

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 $\mathsf{IDTR.Enc}^k(.)$ instead of $\mathsf{IDTR.Enc}^{k-1}(.)$. We call $Adv_{ind,k}^{idtr-wibe}(\mathcal{B})$ the advantage of the adversary $\mathcal{B}$ in Game k. We remark that, in the game w, the adversary $\mathcal{B}$ has zero advantage beacause $\mathcal{B}$ receives two ciphertext of the same message M_0. Therefore, the proof directly holds under the following lemma:

Lemma 14.
$$Adv_{ind,k}^{idtr-wibe}(\mathcal{B}) - Adv_{ind,k-1}^{idtr-wibe}(\mathcal{B}) \leq \epsilon^*,$$
where ϵ^* is the bound on the advantages of the adversaries against WIBE.

Proof. We will construct an anversary $\mathcal{B}'$ that breaks the $\mathsf{IND\text{-}WID\text{-}CPA}$ securrity of the underlying WIBE with an advantage of $Adv_{ind,k}^{idtr-wibe}(\mathcal{B}) - Adv_{ind,k-1}^{idtr-wibe}(\mathcal{B})$.

Setup: The challenger of $\mathcal{B}'$ runs setup algorithm of WIBE 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 key query for an user $id = id_1 \ldots id_l$ in a group ID, $\mathcal{B}'$ sends the query $WID = (ID, id_1, \ldots, id_l)$ (a $(l+1)-$vector) to its challenger and gets a key d_{WID}. $\mathcal{B}'$ passes this key d_{WID} 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 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 $C_b = \mathsf{WIBE}.\mathbf{Enc}(\mathsf{msk}, ID_{S_{i_k}}, M_b)$.

$\mathcal{B}'$ 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:

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

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 queries on ID^*, $\mathcal{B}'$ will not make 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 $|Adv_{ind,k}^{idtr-wibe}(\mathcal{B}) - Adv_{ind,k-1}^{idtr-wibe}(\mathcal{B})|$.

C Proof of Theorem 9

We first assume that there exists an adversary A that breaks the $\mathsf{IND\text{-}GWID\text{-}CPA}$-security of the $\mathsf{Wa\text{-}GWIBE}$ scheme, then we show how to efficiently build another adversary B to break the security of the $\mathsf{Wa\text{-}HIBE}$ scheme. B runs as follow.

- Let $mpk_B = (g_1, g_2, h_1, u'_{1,0}, ..., u'_{L,n})$ be the master public key of the $\mathsf{Wa\text{-}HIBE}$ scheme.
- $\mathcal{B}$ guess a parttern $P = (P_1, ..., P_l)$. The most important thing here is to guess the position of wildcards in each level. $\mathcal{B}$ hope that the ultimate tagetted parttern $P*$ outputted by $\mathcal{A}$ will have the same wildcard with P ($\mathcal{B}$ will abort if this event does not occur).

– $\mathcal{B}$ computes the master key $mpk_A = (g_1, g_2, h_1, u_{1,0}, ..., u_{L,n})$ as follow:

$$u_{i,j} = u'_{i,j} \qquad if \qquad i \in \overline{W}(P)$$

$$u_{i,j} = u'_{i,j} \qquad where \;\; j = 0, ..., ip, \;\; and \;\; j = n - is + 1, ..., n \qquad if \;\; i \in W(P)$$

$$u_{i,j} = g_1^{\alpha_{i,j}} \qquad if \;\; i \in W(P) \;\; and \;\; j = ip + 1, ..., n - is \qquad where \;\; \alpha_{i,j} \xleftarrow{\$} \mathbb{Z}_p$$

– $\mathcal{B}$ runs $\mathcal{A}$ on mpk_A. If A makes a key derivation oracle on input $ID = (ID_1, ..., ID_l)$ then B constructs an identity $ID' = (ID'_1, ..., ID'_l)$ by setting $ID'_i = ID_i$ for each $i \in \overline{W}(P)$, and $ID'_i = ID_{i,ip}0...0ID_{i,is}$ for each $i \in W(P)$.
B queries its key derivation oracle on ID' and receives $(a'_0, ..., a'_l)$ as follow.

$$(a'_0, a'_1, \cdots, a'_l) = \left(h_2 \prod_{i=1}^{l} F_i(ID'_i)^{r_i}, g_1^{r_1}, \cdots, g_1^{r_l}\right)$$

B reconstructs the decryption key for identity ID as follow.

$$a_0 = a'_0 \times \prod_{i \in W(P)}^{j=ip+1,...,n-is; j \in ID_i} a_i'^{\alpha_{i,j}}, \qquad a_i = a'_i \qquad for \qquad i = 1, ..., l$$

We see that $(a_0, ..., a_l)$ is key corresponding to identity ID in Wa-GWIBE because

$$a_0 = a'_0 \times \prod_{i \in W(P)}^{j=ip+1,...,n-is; j \in ID_i} a_i'^{\alpha_{i,j}} = h_2 \times \prod_{i=1}^{l} F_i(ID'_i)^{r_i} \times \prod_{i \in W(P)}^{j=ip+1,...,n-is; j \in ID_i} a_i'^{\alpha_{i,j}}$$

$$= h_2 \times \prod_{i \in \overline{W}(P)} F_i(ID_i)^{r_i} \times \prod_{i \in W(P)} F_i(ID_{i,ip}0...0ID_{i,is})^{r_i} \times \prod_{i \in W(P)}^{j=ip+1,...,n-is; j \in ID_i} a_i'^{\alpha_{i,j}}$$

$$= h_2 \times \prod_{i \in \overline{W}(P)} F_i(ID_i)^{r_i} \times \prod_{i \in W(P)} F_i(ID_i)^{r_i} = h_2 \times \prod_{i=1}^{l} F_i(ID_i)^{r_i}$$

B returns the key $(a_0, ..., a_l)$ to A.

– A outputs the challenge pattern $P^* = (P_1^*, ..., P_l^*)$ and challenge messages m_0, m_1.

– Denotes $\mathsf{cond}(P^*, P)$ the predicate to represent the event that P^* and P have wildcards at exactly same positions. It means that at each level, either P_i^* and P_i do not contain a wildcard, or they contain wildcards with the same start and final positions. If not $\mathsf{cond}(P^*, P)$ then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ is able to construct the corresponding HIBE identity ID^* from P^* as above. $\mathcal{B}$ outputs the challenge identity ID^* and two messages m_0, m_1.
Let $C'^* = (C_1'^*, C_2'^*, C_3'^*)$ be the challenge ciphertext that $\mathcal{B}$ receives in return from its challenger, meaning that C'^* is an encryption of m_b with respect to the identity ID^*, where b is the secret bit chosen at random by the challenger.
B sets

$$C_1^* = C_1'^* \qquad C_3^* = C_3'^* \qquad C_2^* \leftarrow (C_{2,i}^* = C_{2,i}'^*)_{i \in \overline{W}(P)}$$

$$C_4^* \leftarrow (C_{4,ip,is}^* = C_{2,i}'^*, (C_{4,i,j}^* = C_1'^{*\alpha_{i,j}})_{j=ip+1,...,n-is})_{i \in W(P)}$$

– $\mathcal{B}$ runs $\mathcal{A}$ on the input $C^* = (C_1^*, C_2^*, C_3^*, C_4^*)$. If A makes a key derivation oracle query, then $\mathcal{B}$ answer its queries as before. $\mathcal{A}$ outputs a guess β'

– $\mathcal{B}$ outputs β'

Moreover, we easily see that the challenge ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ sent to $\mathcal{A}$ can be seen to be correctly formed when $\mathsf{cond}(P^*, P)$.

To conclude the proof, we notice that the success probability of $\mathcal{B}$ is at least as same as $\mathcal{A}$ when its guess for $W(P^*)$ is correct, ı.g., $\mathsf{cond}(P^*, P)$. We now compute this probability. At each level $\mathcal{B}$ has to guess the start position and the final position of wildcard (note that if the start position is $n+1$ then it means there is no wildcard at this level). The probability of a good guess at a level is $1/n^2$. Moreover, there are l levels, therefore the probability that A guesses $W(P^*)$ correctly is $1/n^{2 \times l}$.

D The Method of Jin and Lotspiech

Recently , [JL09] proposes a new method to defend against pirate evolution attacks in subset difference method but their method decreases the efficiency of the original scheme. In subset difference method of [NNL01] each user u possesses a set of decryption keys (long live keys) associated to $S_{i,j}$ $(i, j \in [1, \cdots, logN])$ where u is a descendant of v_i but not v_j.
Given the private keys of t traitors, the master pirate box decoder B spawns successively a sequence of pirate boxes decoders $B_1, B_2, \cdots$ by associating each long live key of a set $S_{i,j}$ to each pirate box decoder. If given the set of decryption keys of a traitor $u \in S_{i,j}$ (u is not a descendant of v_j) then master pirate box decoder B can spawn a series of pirate boxes decoder by using the long live keys of a series sets $S_{c,j}$, where v_c is a node on the path from v_i to v_j.

In [JL09] the method they proposed is very simple, they only level down the key encryption $K_{S_{i,j}}$ to the desire level by, first, splitting $K_{S_{i,j}} = K_{S_{i,j_1}}, K_{S_{j_1,j_2}}, \cdots, K_{S_{j_h},j}$ where $j_1, \cdots, j_h$ are nodes on the path from i to j, then continuing to level down each subset $K_{S_{i,j_1}}, K_{S_{j_1,j_2}}, \cdots, K_{S_{j_h},j}$ to the desire level as long as the combination of these subsets covers all users in the set $S_{i,j}$. They proved that if the tracing algorithm wants to detect and disable a traitor in only b generations of pirate decoder, each subset needs to be expanded into multi-subsets; the overall subsets are expanded is linear in $N^{1/b}$. Therefore, the ciphertext size will magnify to a factor of $O(N^{1/b})$.

By this way they can restrain master pirate box B from spawning B_i. In exchange, however, the size of ciphertext will be trade off. If their scheme resists completely to pirate evolution, the size of ciphertext in their scheme must be linear in the number of the user. They stress the case that when generations of clone decoder is 2, the efficiency of the system is acceptable in the practical. In this case, however, the length of the ciphertext and private key of their scheme are still less efficient than ours $(\sqrt{N}, log^2(N)$ compare to $O(r \log(\frac{N}{r}) log(N)), O(1))$.

E The Methods in [DdP11] and [ZZ11]

The method in [DdP11]: The main idea in [DdP11] is, for each subset S_i in CS scheme (or $S_{i,j}$ in SD scheme) they choose uniform at random a secret t-degree polynomial P_i where t is the maximum number of users collude. Each user u belongs to subset S_i in CS scheme (or $S_{i,j}$ in SD scheme) receives the pair $(i, P_i(I_u))$.
To broadcast, for each subset S_i in CS scheme (or $S_{i,j}$ in SD scheme) they use the secret key K_i (or $K_{i,j}$) that based on $P_i(0)$ ($P_i(0)$ can be recovered by using $t+1$ different values of polynomial P_i) to encrypt the session key K, broadcast this ciphertext and t different values which based on $P_i(I_1), \ldots, P_i(I_t)$ where $I_1, \ldots, I_t$ are different all I_u. To decrypt, each user belongs to a subset S_i can combine these t different values with its value to recover K_i (or $K_{i,j}$) and decrypt.
By this way they can restrain the user from publishing his private key to public domain, because if do that, user immediately reveal his sensitive information.
The drawback of their scheme is if more than t user collude in subset S_i, then they can compute $P_i(0)$ and publish this value to the public domain, so all other users can decrypt. To against this drawback, they have to increase the value of t, however in exchange, their schemes' parameters size become inefficient (their schemes' ciphertext size are bigger than NNL schemes' ciphertext size a multiplied factor of t). Compare with our best scheme, the private key size of our scheme is better (the private key size of their scheme are the same as NNL schemes while the private key size of our scheme is constant). Our ciphertext size is also better than theirs in the case their schemes resist full collusion of users (to resist full collusion of users, their schemes' ciphertext size must be linear in the number of the users in the system).

The method of [ZZ11]: [ZZ11] also proposed a new method to defend against pirates 2.0 in codebase schemes. However, their method is similar to the method in [ADML$^+$07] except they split the second level in [ADML$^+$07] into L levels (L is the length of code). Note that like [ADML$^+$07], their scheme does not support revocability.
Therefore, the comparison between their scheme's parameters size with our schemes' parameters size is the same as the comparison between [ADML$^+$07]'s parameters size with our schemes' parameters size.

F Concrete Construction of IDTR From Waters' WIBE-2

F.1 Construction

Let $\mathbb{G}, \mathbb{G}_T$ be multiplicative groups of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \leftarrow \mathbb{G}_T$.

Setup(1^k)**:** The trusted authority chooses randomly generators $g_1, g_2, u_{1,0}, \cdots, u_{1,n}, u_{2,0}, u_{2,1}, \cdots, u_{n+1,0}, u_{n+1,1} \xleftarrow{\$} \mathbb{G}^*$ and a random value $\alpha \xleftarrow{\$} \mathbb{Z}_p$, where $n+1$ is the maximum hierachy depth. For a system of maximum N users, we define $n = \log(N)$. Next, it computes $h_1 \leftarrow g_1^\alpha$ and $h_2 \leftarrow g_2^\alpha$.
The master public key is $mpk = (g_1, g_2, h_1, u_{1,0}, \cdots, u_{1,n}, u_{2,0}, u_{2,1}, \cdots, u_{n+1,0}, u_{n+1,1}, F)$ and the master secret key is $msk = h_2$.
For each level $i = 1, 2, \ldots, n+1$, a function $F_i : \{0,1\}^n \to \mathbb{G}^*$ is defined as:

$$F_i(ID_i) = u_{i,0} \prod_{j \in ID_i} u_{i,j}$$

The Setup also defines a symmetric encryption scheme (E_K, D_K) to encrypt the message, as described in Section 3.2.

KeyDer(msk, ID, id) A user $id = id_2 \cdots id_{n+1}$ where $id_i \in \{0,1\}$ and $i = 2, \cdots, n+1$, in a group ID is given the private key $d_{ID,id}$ which is computed as

$$d_{ID,id} = (a_0, a_1, a_2, \cdots, a_{n+1}) = \left(h_2 \times F_1(ID)^{r_1} \times \prod_{i=2}^{n+1} F_i(id_i)^{r_i}, g_1^{r_1}, \cdots, g_1^{r_{n+1}}\right)$$

Enc$(\mathsf{mpk}, ID, \mathcal{R}_{ID}, M)$ The encryption algorithm takes as input a 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.
The revocation works as in the complete subtree scheme and similar to the generic case where the users in $\mathcal{N}_{ID} \backslash \mathcal{R}_{ID}$ are partitioned into disjoint subsets S_{i_1}, ..., S_{i_w}.
Each subset S_{i_j}, $1 \le j \le w$, is associated to an $n+1$ vector identity $ID_{S_{i_j}} = (ID, id_{i_j,2}, \cdots, id_{i_j,k}, *, *.., *)$ where $id_{i_j,2}, \cdots, id_{i_j,k}$ is the path from the root ID to the node S_{i_j} and the number of wildcards $*$ is $n - k + 1$.
For each identity $ID_{S_{i_j}} = (ID, id_{i_j,2}, \cdots, id_{i_j,k}, *, *.., *)$, the encryptor chooses $t \xleftarrow{\$} \mathbb{Z}_p$ and computes $C^j = (ID_{S_{i_j}}, C_1^j, C_2^j, C_3^j, C_4^j)$ as follow:

$$C_1^j \leftarrow g_1^t \qquad C_2^j \leftarrow (C_{2,1}^j = F_1(ID)^t, (C_{2,z}^j = F_z(id_{i_j,z})^t)_{2 \le z \le k})$$
$$C_3^j \leftarrow K \cdot \hat{e}(h_1, g_2)^t \qquad C_4^j \leftarrow (C_{4,z,y}^j = u_{z,y}^t)_{z=k+1,\ldots,n+1;\ y=0,1}$$

And outputs the ciphertext

$$C = \langle [i_1, \cdots, i_w][C^1, \cdots, C^w,], E_K(M) \rangle$$

Dec$(d_{ID,id}, C)$**:** When a user receives the ciphertext C as above. Firstly, it finds j such that $id \in S_{i_j}$ (in case $id \in \mathcal{R}_{ID}$ the result is null). Secondly, uses private key $d_{ID,id}$ to decrypt C^j to obtain K. Finally, the user computes $D_K(E_K(M))$ to recover the message M.
We now explain how the user can decrypt C^j. It first computes $C_2^{\prime j} = (C_{2,i}^{\prime j})_{i=1,\ldots,n+1}$as:

$$C_{2,1}^{\prime j} = C_{2,1}^j, \quad C_{2,z}^{\prime j} = F_z(id_z)^t \leftarrow \begin{cases} C_{2,z}^j & \text{if } 2 \le z \le k \\ C_{4,z,0}^j \cdot \prod_{y \in id_z} C_{4,z,y}^j & \text{if } k+1 \le z \le n+1 \end{cases}$$

then computes

$$C_3^j \cdot \frac{\prod_{i=1}^{n+1} \hat{e}(a_i, C_{2,i}^{\prime j})}{\hat{e}(C_1, a_0)} = K$$

After obtaining K, user id computes $D_K(E_K(M))$ to obtain and output M.

Trace$^{\mathbb{D}}(\mathsf{msk}, ID)$**:** Our tracing algorithm is similar to the one in [NNL01]. Takes as input msk, ID, an illegal decryption box $\mathbb{D}$, returns either a subset consisting the traitors or a new partition of $\mathcal{N}_{ID} \backslash \mathcal{R}_{ID}$ that renders the illegal decryption box useless.

F.2 Security Analysis

We recall theorem 5.1 from [ACD+06] on the security of Wa-WIBE.

Theorem 15. *[ACD+06] If the* Wa-HIBE *scheme of depth l is (t, q_K, ϵ)* IND-ID-CPA*-secure, then the* Wa-WIBE *scheme of depth l is (t', q'_K, ϵ')* IND-WID-CPA*-secure with*

$$t' \leq t - l \cdot n(1 + q_K) \times t_{exp}, \quad q'_K \leq q_K, \quad \epsilon' \leq \epsilon \cdot 2^l$$

where n is the length of identities at each level, t_{exp} is the time taken by a group exponentiation.

Waters' WIBE *used in our trace and revoke system.* In our construction of a trace and revoke system, we only need a simplified $l+1$-level WIBE, that we call WIBE-2. In WIBE-2, each identity has the form $(ID, id_1, \ldots, id_l)$ where ID is the group identity of size n and each id_i, $1 \leq i \leq l$, is a bit 0 or 1. This means that in the last l levels, the size of identities are 1. The bound about t' in Theorem 15 becomes $t' \leq t - ((n(1 + q_K) + l(1 + q_K)) \times t_{exp}$.

Moreover, the pattern P used in our trace and revoke system also has a particular form of $ID_{S_{i_j}} = (ID, id_{i_j,1}, \ldots, id_{i_j,k}, *, *, \ldots, *)$. This means that all the wildcards are grouped at the end. In the proof of Theorem 15, the factor 2^l comes from the guess of the positions of wildcards. In our case, we only need to guess the position of the first wildcard, all the other wildcards automatically follow. Therefore, in our case: $\epsilon' \leq \epsilon \cdot l$

We can finally deduce, from Theorem 2, the security of our scheme when instantiated with the Waters-HIBE:

Theorem 16. *If the* Wa-HIBE *of depth l is (t, q_K, ϵ)* IND-ID-CPA*-secure, then the* IDTR *scheme based on* WIBE-*2 is (t^*, q^*, ϵ^*)* IND-CPA*-secure with*

$$t^* \leq (t - ((n(1 + q_K) + l(1 + q_K)) \times t_{exp})/(r \log(\frac{N}{r})), \quad q^*_K \leq q_K, \quad \epsilon^* \leq r \log(\frac{N}{r}) \times \epsilon \cdot l.$$

G Construction of IDTR from 2-Level Wa-GWIBE (2level-Wa-GWIBE-IDTR)

Let $\mathbb{G}, \mathbb{G}_T$ be multiplicative groups of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \leftarrow \mathbb{G}_T$.

Setup(1^k)**:** The trusted authority chooses randomly generators $g_1, g_2, u_{1,0}, \cdots, u_{1,n}, u_{2,0}, \cdots, u_{2,n} \xleftarrow{\$} \mathbb{G}^*$ and a random value $\alpha \xleftarrow{\$} \mathbb{Z}_p$. n is the size of identities at each level. For a system of maximum N users, we define $n = \log(N)$.

Next, it computes $h_1 \leftarrow g_1^\alpha$ and $h_2 \leftarrow g_2^\alpha$. The master secret key is $msk = h_2$ and the master public key is $mpk = (g_1, g_2, h_1, u_{1,0}, \cdots, u_{1,n}, u_{2,0}, \cdots, u_{2,n})$. For each level $i = 1, 2$, a function $F_i : \{0,1\}^n \to \mathbb{G}^*$ is defined as:

$$F_i(ID_i) = u_{i,0} \prod_{j \in ID_i} u_{i,j}$$

The Setup also defines a symmetric encryption scheme (E_K, D_K) to encrypt the message, as described in Section 3.2.

KeyDer(msk$, ID, id$) A user id in a group ID is given the private key $d_{ID,id}$ which is computed as

$$d_{ID,id} = (a_0, a_1, a_2) = \left(h_2 \times F_1(ID)^{r_1} \times F_2(id)^{r_2}, g_1^{r_1}, g_1^{r_2}\right)$$

where $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$.

Enc(mpk$, ID, \mathcal{R}_{ID}, M$) The encryption algorithm takes as input a 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.

The revocation works as in the complete subtree scheme and similar to the generic case where the users in $\mathcal{N}_{ID} \backslash \mathcal{R}_{ID}$ are partitioned into disjoint subsets S_{i_1} , ..., S_{i_w}. Each subset S_{i_j}, $1 \leq j \leq w$, is associated to a 2 dimension vector identity $ID_{S_{i_j}} = (ID, id_{i_j} || *)$.

The encryptor randomly chooses a session key K, then encrypts w times K by using 2-Level $\mathsf{Wa\text{-}GWIBE}$ on the identities of subgroups S_{i_j} $(j = 1, ..., w)$.

For each identity $ID_{S_{i_j}} = (ID, id_{i_j}||*)$ where the size of id_{i_j} is ip, the encryptor chooses $t \overset{\$}{\leftarrow} \mathbb{Z}_p$ and computes $C^j = (ID_{S_{i_j}}, C^j_1, C^j_2, C^j_3, C^j_4)$, where

$$C^j_1 \leftarrow g^t_1 \qquad C^j_2 \leftarrow (C^j_{2,1} = F_1(ID)^t) \qquad C^j_3 \leftarrow K \cdot \hat{e}(h_1, g_2)^t$$

$$C^j_4 = (C^j_{4,ip,n} = F_2(id_{i_j}||00\dots0)^t \ , \ (C^j_{4,2,z} = u^t_{2,z})_{z=ip+1,\dots,n})$$

Finally, the encryptor outputs the ciphertext

$$C = \langle [i_1, \cdots, i_w][C^1, \cdots, C^w], E_K(M)\rangle$$

Dec$(d_{ID,id}, C)$**:** When a user receives the ciphertext C as above. Firstly, it finds j such that $id \in S_{i_j}$ (in case $id \in \mathcal{R}_{ID}$ the result is null). Secondly, uses private key $d_{ID,id}$ to decrypt C^j to obtain K. Finally, the user computes $D_K(E_K(M))$ to recover the message M.
The user id decrypts the ciphertext $C^j = (ID_{S_{i_j}}, C^j_1, C^j_2, C^j_3, C^j_4)$ by computing $C'^j_2 = (C'^j_{2,i})_{i=1,2}$ as

$$C'^j_{2,1} = C^j_{2,1}, \quad C'^j_{2,2} = C^j_{4,ip,n} \times \prod_{z=ip+1,\dots,n}^{z\in id} C^j_{4,2,z} = F_2(id)^t$$

then computes

$$C^j_3 \cdot \frac{\prod_{i=1}^2 \hat{e}(a_i, C'^j_{2,i})}{\hat{e}(C^j_1, a_0)} \quad = \quad K \cdot \hat{e}(h_1, g_2)^t \cdot \frac{\hat{e}(g^{r_1}_1, F_1(ID)^t) \cdot \hat{e}(g^{r_2}_1, F_2(id)^t)}{\hat{e}(g^t_1, h_2 \cdot F_1(ID)^{r_1} \cdot F_2(id)^{r_2})} \quad = \quad K$$

Trace$^{\mathbb{D}}(\mathsf{msk}, ID)$**:** Our tracing algorithm is similar to the one in [NNL01]. Takes as input msk, ID, an illegal decryption box $\mathbb{D}$, returns either a subset consisting the traitors or a new partition of $\mathcal{N}_{ID}\backslash\mathcal{R}_{ID}$ that renders the illegal decryption box useless.