Identity-Based Traitor Tracing

Michel Abdalla[1], Alexander W. Dent[2], John Malone-Lee[3], Gregory Neven[1,4], Duong Hieu Phan[5], and Nigel P. Smart[3]

[1] Département d'Informatique, Ecole Normale Supérieure,
45 Rue d'Ulm, 75230 Paris Cedex 05, France.
Email: {Michel.Abdalla,Gregory.Neven}@ens.fr

[2] Information Security Group, Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom.
Email: a.dent@rhul.ac.uk

[3] Department Computer Science, University of Bristol,
Woodland Road, Bristol, BS8 1UB, United Kingdom.
Email: {malone,nigel}@cs.bris.ac.uk

[4] Department of Electrical Engineering, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
Email: Gregory.Neven@esat.kuleuven.be

[5] France Télécom R&D, 38-40 rue du Général Leclerc,
92794 Issy les Moulineaux Cedex 9, France.
Email: duonghieu.phan@orange-ftgroup.com

Abstract. We present the first identity-based traitor tracing scheme. The scheme is shown to be secure in the standard model, assuming the bilinear decision Diffie-Hellman (DBDH) is hard in the asymmetric bilinear pairing setting, and that the DDH assumption holds in the group defining the first coordinate of the asymmetric pairing. Our traitor tracing system allows adaptive pirates to be traced. The scheme makes use of a two level identity-based encryption scheme with wildcards (WIBE) based on Waters' identity-based encryption scheme.

1 Introduction

In 1984 Shamir proposed the concept of identity-based cryptography [15]. However, it took nearly twenty years for the problem of designing an efficient method to implement identity-based encryption (IBE) to be solved. In 2000 and 2001 respectively Sakai, Ohgishi and Kasahara [13] and Boneh and Franklin [6] proposed IBE schemes based on elliptic curve pairings. Also, in 2001 Cocks proposed a system based on the quadratic residuosity problem [10].

The fourth author is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO). The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Identity-based encryption is often justified as a useful technology by its possible use in an e-mail application. However, many people, whilst having a small set of e-mail identities, often belong to a larger set of e-mail groups. An e-mail group, or shared address, is an e-mail address which allows the sender to send a message to a large number of individual e-mail addresses without needing to know the actual individual addresses. Using existing identity-based encryption techniques one can easily implement such a scheme by giving each member of the e-mail group the same ID-private key. Thus all members of the group will share the same private key.

A common business model in PKI world is that the certificate authority charges for each certificate, or block of certificates, issued. In the ID-based world this model corresponds to the trust authority charging for each private key, or block of private keys. However, in our group e-mail example this would mean that the trust authority would only be able to charge for one private key for the whole group, since as soon as one person had the private key they could share it with the other members of the group. What is needed is a disincentive for the group members to collaborate in this manner.

A similar situation occurs in the traditional symmetric or public key setting in broadcast encryption. Here one solves the associated problem by using a traitor tracing scheme, which allows any person (or set of colluding people) who creates a new decryption device, or key, to be traced. Thus combining the above ideas together we see that there is a possible need for an identity-based traitor tracing scheme.

Surprisingly since the invention of identity-based cryptography by Shamir [15] in 1984, no one seems to have considered this issue. Thus in this paper we present the first identity-based traitor tracing scheme. Our scheme is based on the Waters' WIBE from [1], which is based on Waters' identity-based encryption scheme [17]. A WIBE is a variant of a hierarchical IBE (HIBE) scheme in that it encrypts to an identity string which is defined on various layers. However, unlike a HIBE, which allows only a single recipient, a WIBE allows one to encrypt to a string which is "wildcarded" on a given set of levels. A WIBE allows one to target a ciphertext at a given group of users by applying the appropriate wildcards.

Our construction is relatively simple: we use a two level WIBE in which the first level represents the name of the group and the second level represents the unique index of a user. This allows e-mails to be addressed to the entire group via the use of a wildcard in the second level. Group membership is 'policed' by the trust authority, which only releases a decryption key to a user if the user is entitled to decrypt messages sent to a particular group. The subtlety of our construction is in the construction of a traitor tracing algorithm.

We prove that our scheme protects the confidentiality of encrypted messages against passive attackers in the standard model, and show that it allows traitor tracing against an adaptive traitor.

Unfortunately, our scheme is not practical due to the combination of Waters' IBE and collusion secure codes [8], which results in infeasibly large public key and ciphertext sizes. Thus we leave the construction of a truly efficient identity-based traitor tracing scheme, even in the random oracle model [3], as an open problem. In addition we leave as open the problem of creating a scheme which allows a greater number of key extraction queries by the pirate than ours allows. Furthermore, our scheme does not protect against pirate decoder manufacturers mounting chosen-ciphertext attacks, however this later stronger pirate has not been considered in the public-key setting either.

2 Preliminaries

2.1 Notation

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers and $\{0,1\}^*$ the set of all bit strings. If $k \in \mathbb{N}$ then $\{0,1\}^k$ is the set of bit strings of length k and 1^k is the string of k ones. If $\mathcal{A}$ is a randomized algorithm, then $y \overset{\$}{\leftarrow} \mathcal{A}^O(x)$ denotes the assignment to y of the output of $\mathcal{A}$ when run on input x with fresh random coins and with access to oracle O; we write $y \leftarrow \mathcal{A}^O(x)$ if $\mathcal{A}$ is deterministic. If S is a finite set, then $x \overset{\$}{\leftarrow} S$ denotes the random generation of an element $x \in S$ using the uniform distribution. A function $\nu : \mathbb{N} \to [0,1]$ is said to be *negligible* if for all $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $\nu(k) < k^{-c}$ for all $k > k_c$. It is said to be *non-negligible* if there exists a $c \in \mathbb{N}$ such that $\nu(k) > k^{-c}$ for all $k \in \mathbb{N}$.

2.2 Computational Assumptions

Our scheme employs asymmetric pairings, which we now recall. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_\mathrm{T}$ denote three finite multiplicative abelian groups of prime order $p > 2^k$. Let g and h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be an efficiently computable isomorphism such that $\psi(h) = g$. We assume that there exists an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathrm{T}$, meaning that for all $a, b \in \mathbb{Z}_p$ (1) $\hat{e}(g^a, h^b) = \hat{e}(g,h)^{ab}$, (2) $\hat{e}(g^a, h^b) = 1$ iff $a = 0$ or $b = 0$, and (3) $\hat{e}(g^a, h^b)$ is efficiently computable.

The advantage of an algorithm $\mathcal{A}$ in solving the *computational bilinear Diffie–Hellman (CBDH)* problem in $\mathbb{G}_2$ is defined as

$$\mathbf{Adv}^{\mathrm{cbdh}}_{\mathcal{A},\mathbb{G}_2}(k) \;=\; \Pr\Big[Z = \hat{e}(g,h)^{xyz} \;:\; x,y,z \overset{\$}{\leftarrow} \mathbb{Z}_p \;;\; Z \overset{\$}{\leftarrow} \mathcal{A}(h^x, h^y, h^z)\Big] \;.$$

The advantage of $\mathcal{A}$ in solving the decisional variant of this problem, called the *decisional bilinear Diffie–Hellman (DBDH)* problem in $\mathbb{G}_2$, is

$$\mathbf{Adv}^{\mathrm{dbdh}}_{\mathcal{A},\mathbb{G}_2}(k) = \Bigg| \Pr\Big[\mathcal{A}(h^x, h^y, h^z, Z) = 1 \;:\; x,y,z \overset{\$}{\leftarrow} \mathbb{Z}_p \;;\; Z \leftarrow \hat{e}(g,h)^{xyz}\Big] - \Pr\Big[\mathcal{A}(h^x, h^y, h^z, Z) = 1 \;:\; x,y,z \overset{\$}{\leftarrow} \mathbb{Z}_p \;;\; Z \overset{\$}{\leftarrow} \mathbb{G}_\mathrm{T}\Big] \Bigg| \;.$$

We say that the CBDH and DBDH problems in $\mathbb{G}_2$ are *hard* if the respective advantages are negligible functions in k for all algorithms $\mathcal{A}$ with running time polynomial in k.

We also require that the DDH problem in $\mathbb{G}_1$ is hard, namely we require that for all algorithms $\mathcal{A}$, with running time polynomial in k, the following advantage is a negligible function in k,

$$\mathbf{Adv}^{\text{xddh}}_{\mathcal{A},\mathbb{G}_1}(k) = \Bigg| \Pr\left[\mathcal{A}(g^x, g^y, Z) = 1 \ :\ x, y \xleftarrow{\$} \mathbb{Z}_p \ ;\ Z \leftarrow g^{xy}\right] - \Pr\left[\mathcal{A}(g^x, g^y, Z) = 1 \ :\ x, y \xleftarrow{\$} \mathbb{Z}_p \ ;\ Z \xleftarrow{\$} \mathbb{G}_1\right] \Bigg| .$$

Note that if the DDH problem in $\mathbb{G}_1$ is hard, then there cannot exist a computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ and thus we must be working in the asymmetric pairing setting. The assumption that the DDH problem is hard in $\mathbb{G}_1$ is referred to as the external DDH problem (XDDH) and has been used before in [2, 4, 14].

3 Identity-Based Traitor Tracing

3.1 Syntax

In this section we will describe the general model for an identity-based traitor tracing scheme. Broadcast groups are referred to by an identity string $ID \in \{0,1\}^*$, individual users are referred to by an index $i \in \mathbb{N}$. To make user i member of the group ID, the trusted key distribution centre provides it with a personal decryption key $d_{ID,i}$. Anyone can encrypt a message to the general group ID such that all individual users belonging to the group can recover the message.

Formally, an identity-based traitor tracing scheme $\mathcal{IBTT}$ consists of five polynomial-time algorithms:

- A randomised key generation algorithm $\mathcal{G}(1^k)$ taking as input the security parameter k. This algorithm generates a set of domain parameters consisting of a master public key mpk and a master secret key msk.
- A key extraction algorithm $\mathcal{X}(msk, ID, i)$ which given the master secret key msk, a group identity $ID \in \{0,1\}^*$ and a user index i generates a user secret key $d_{ID,i}$. This algorithm could be probabilistic.
- A probabilistic encryption algorithm $\mathcal{E}(mpk, ID, \mathfrak{m})$ which on input of the master public key mpk, a group identity ID and a message $\mathfrak{m}$ outputs a ciphertext C.
- A decryption algorithm $\mathcal{D}(d_{ID,i}, C)$ which on input of a user secret key $d_{ID,i}$ and a ciphertext C outputs a plaintext message $\mathfrak{m}$, or $\perp$ to indicate a decryption error.
- A traitor tracing algorithm $\mathcal{T}^{\mathbb{D}}(msk, ID)$ 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 a set of user identifiers (called "traitors") $T \subset \mathbb{N}$.

An identity-based traitor tracing scheme whose tracing algorithm takes as input *mpk* instead of *msk* is said to be *publicly-traceable*, since then anyone can execute the tracing algorithm. We shall assume that all "pirate" decryption boxes are resettable [11], meaning that they retain no state between decryptions. In particular, pirate boxes cannot self-destruct.

For correctness we require that $\mathcal{D}(d, \mathcal{E}(mpk, ID, \mathfrak{m})) = \mathfrak{m}$ with probability one for all $k \in \mathbb{N}$, $ID, \mathfrak{m} \in \{0,1\}^*$, $i \in \mathbb{N}$, $(mpk, msk) \xleftarrow{\$} \mathcal{G}(1^k)$ and $d \xleftarrow{\$} \mathcal{X}(msk, ID, i)$.

3.2 Secrecy

We require that our ID-based traitor tracing scheme is semantically secure in the presence of adaptive adversaries who have access to a key extraction oracle and, in a chosen-ciphertext attack, a decryption oracle. These are standard notions in ID-based cryptography first introduced in [6]. The extension to the setting we have here is immediate, but for completeness we clarify it here.

Secrecy is defined by a two-stage game. The challenger first runs the key generation algorithm to generate a master key pair $(mpk, msk) \xleftarrow{\$} \mathcal{G}(1^k)$. The master public key *mpk* is passed to the adversary. In the first stage of the game the adversary has access to a key extraction oracle $\mathcal{X}(msk, \cdot, \cdot)$, which it can query on arbitrary pairs (ID, i) of group identities ID and user indices i. In a chosen-ciphertext attack, the adversary can also has access to a decryption oracle $\mathcal{D}(\mathcal{X}(msk, \cdot, \cdot), \cdot)$ from which it can obtain the decryption of any ciphertext C using the key to any pair (ID, i). The first stage ends when the adversary outputs two messages of equal length $\mathfrak{m}_0$ and $\mathfrak{m}_1$, plus a challenge group identity ID^*.

The challenger then selects a bit b and encrypts $\mathfrak{m}_b$ under the group identity ID^* to form the challenge ciphertext $C^* \leftarrow \mathcal{E}(mpk, ID^*, \mathfrak{m}_b)$. The challenge ciphertext is returned to the adversary for the second stage of the game. In this second stage the adversary can perform further queries to its oracles. At the end of the second stage the adversary outputs its guess b' as to the bit b. The adversary wins the game if $b = b'$, if ID^* never appeared in any of the key extraction oracle queries, and, in a chosen-ciphertext attack, if C^* was never submitted to the decryption oracle with group identity ID^*.

The advantage $\mathbf{Adv}^{\text{ind-id-cpa}}_{\mathcal{A},\mathcal{IBTT}}(k)$, respectively $\mathbf{Adv}^{\text{ind-id-cca}}_{\mathcal{A},\mathcal{IBTT}}(k)$, of an adversary $\mathcal{A}$ in breaking the indistinguishability of scheme $\mathcal{IBTT}$ 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.

3.3 Traceability

We extend the notion of traceability defined for the public key setting in [7] to the identity-based setting. We provide definitions for both chosen-plaintext and chosen-ciphertext attack; our scheme however is only proved secure in the chosen-plaintext setting. We note that to our knowledge there is no public-key traitor tracing system which has been considered in the presence of (the natural analogue of) chosen-ciphertext attacks against the traceability property.

Let $k, c \in \mathbb{N}$ be two security parameters associated to the experiment. The challenger first generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathcal{G}(1^k)$ and gives mpk to the adversary. The adversary has access to a key extraction oracle $\mathcal{X}(msk, \cdot, \cdot)$ to which it can submit pairs (ID, i) of its choosing. In a chosen-ciphertext attack, it can also perform queries to a decryption oracle $\mathcal{D}(\mathcal{X}(msk, \cdot, \cdot), \cdot)$ specifying a group identity ID, a user index i and an arbitrary ciphertext C as in the above secrecy game. The adversary terminates by outputting a group identity ID^* and a pirate decoder $\mathbb{D}$, which is the description of a probabilistic circuit that takes as input ciphertexts and outputs messages. The challenger then runs the tracing algorithm with black-box access to $\mathbb{D}$ to obtain a set of user identifiers $S \xleftarrow{\$} \mathcal{T}^{\mathbb{D}}(msk, ID^*)$.

By modelling the pirate decoder as a probabilistic circuit, we assume that the decoder is *resettable* or *stateless* [11] in that it does not retain information from previous decryptions, and in particular that it cannot self-destruct. Thus, when being subjected to a series of tracing queries, the pirate decoder responds to each query as if it were the first.

If we let T denote the set of user indices i that the adversary submitted to the key extraction oracle in combination with the group identity ID^*, then we say that the adversary wins the game if the following conditions hold:

– The decryption box decrypts a non-negligible fraction of random ciphertexts encrypted under the group identity ID^*, i.e. for random messages $\mathfrak{m}$ we have that $\Pr[\mathbb{D}(\mathcal{E}(mpk, ID^*, \mathfrak{m})) = \mathfrak{m}] \geq \delta(k)$ where $\delta(k)$ is a non-negligible function and where the probability is taken over the random choice of $\mathfrak{m}$ and over the random coins of the encryption algorithm $\mathcal{E}$ and the pirate box $\mathbb{D}$.
– Either $S = \emptyset$ or $S \not\subseteq T$.
– $\mathcal{A}$ queried the key extraction oracle for at most c different user indices i. We do not restrict the number of different group identities ID for which $\mathcal{A}$ can obtain keys for each of these users (apart from being polynomial in k of course). This reflects that colluding users can use all their decryption keys to construct the pirate box, not just the key corresponding to ID^*. It also means that the number of different groups a single user subscribes to is *not* limited by c.
– In the chosen-ciphertext variant there are no restrictions on $\mathcal{A}$'s queries to the decryption oracle.

The advantage $\mathbf{Adv}^{\text{tra-id-cpa[c]}}_{\mathcal{A},\mathcal{IBTT}}(k)$, respectively $\mathbf{Adv}^{\text{tra-id-cca[c]}}_{\mathcal{A},\mathcal{IBTT}}(k)$, of $\mathcal{A}$ in breaking the traceability of the scheme $\mathcal{IBTT}$ is defined as its probability of winning the above game. We say that $\mathcal{IBTT}$ is c-TRA-ID-CPA, respectively c-TRA-ID-CCA secure, if this advantage is a negligible function in k for all adversaries $\mathcal{A}$ with running time polynomial in k.

The above definition is essentially a *full access* model. One can, following [5] and [7], define a *minimal access* model in which the oracle available to the tracing algorithm only outputs whether the decoder successfully decrypted the input ciphertext or not, but does not give it the resulting plaintext.

4 The Scheme

Our scheme makes use of the two-level WIBE scheme [1] based on Waters' HIBE scheme [17]. We assume that group identities *ID* are given by strings of length n_1. As user identifiers we associate to each user an element of a code. The mapping between individual users, their indices and their codewords is maintained by the trust authority. In practice the code will be a (c, N, ϵ)-collusion secure code [8], where N is the maximum number users in the system, c is the maximum number of colluders our tracing algorithm can tolerate, and ϵ is the probability of error that a colluder is not traced. A (c, N, ϵ) collusion secure code can be produced using codewords of size $\ell = O(c^2(\log(N) + \log(1/\epsilon)))$ over an alphabet of size $s = 2$ [16]. Our use of collusion secure codes will result in a scheme which is not publicly traceable, since the tracing algorithm for collusion secure codes requires secret randomness.

Before giving a more precise definition of collusion-secure codes, we need to introduce some additional notation. Let Σ be a symbol alphabet of size $|\Sigma| = s$. If $x = x_1 \dots x_\ell \in \Sigma^\ell$ is a string of ℓ symbols and $I = \{1 \leq i_1 < \dots < i_n \leq \ell\}$ is a set of indices, then $x|_I$ is the substring $x_{i_1} \dots x_{i_n}$ containing only those symbols of x at positions in I. Let $W = \{w_1, \dots, w_c \in \Sigma^\ell\}$ be a set of symbol strings, and let I be the set of all positions where all strings in W are equal, i.e. I is the maximal set such that $w_1|_I = w_2|_I = \dots = w_c|_I$. Then the *feasible set* of W is defined as the set of all strings that are equal to $w_1, \dots, w_c$ at positions in I, i.e.

$$\mathrm{FS}(W) \; = \; \{x \in \Sigma^\ell \; : \; x|_I = w_1|_I = \dots = w_c|_I\} \; .$$

A (c, N, ϵ) collusion-secure code of length ℓ over alphabet Σ consists of a set $\mathbb{C}$, called the *codebook*, of indexed codewords $w_r^{(i)}$ for $1 \leq i \leq N$ and $r \in \{0,1\}^\rho$, and a *tracing algorithm* $\mathcal{T}_\mathbb{C}$. These are such that for all collusions $C \subseteq \{1, \dots, N\}$ of size at most c, $W = \{w_r^{(i)} : i \in C\}$, and for all (unbounded) algorithms $\mathcal{A}$ it holds that

$$\Pr\left[\mathcal{T}_\mathbb{C}(x, r) \in C \mid x \in \mathrm{FS}(W);\; x \xleftarrow{\$} \mathcal{A}(W);\; r \xleftarrow{\$} \{0,1\}^\rho\right] \; > \; 1 - \epsilon \; ,$$

where the probability is taken over the choice of r and the random coins of $\mathcal{T}_\mathbb{C}$ and $\mathcal{A}$. Our scheme uses codewords as "identity strings". This presents a small problem: the definition insists that the set C is chosen *before* $\mathcal{A}$'s execution; whereas, we will allow the adversary to chose the set C adaptively via key extraction queries. We solve this problem by introducing a randomly chosen permutation on $\{1, 2, \dots, N\}$, denoted $\pi \xleftarrow{\$} \mathit{Perm}(N)$ (or if it is desired for efficiency a pseudo-random permutation). We associate the codeword $w_r^{(\pi(i))}$ with the i-th user. It is therefore sufficient that

$$\Pr\left[\mathcal{T}_\mathbb{C}(x, r) \in C \;\middle|\; \begin{array}{c} x \in \mathrm{FS}(W);\; x \xleftarrow{\$} \mathcal{A}(W) \\ C \xleftarrow{\$} \mathcal{P}(\mathbb{C}, c, r);\; r \xleftarrow{\$} \{0,1\}^\rho \end{array}\right] \; > \; 1 - \epsilon \; ,$$

where $\mathcal{P}(\mathbb{C}, c, r)$ is the set of subsets of $\{w_r \in \mathbb{C}\}$ of size c.

For non-binary alphabets, we use the natural encoding of symbols as bit strings of length $\lceil \log_2 s \rceil$, so that codewords are represented by bit strings of length $n_2 = \lceil \log_2 s \rceil \cdot \ell$.

To set up the scheme we define two sets V_1 and V_2 of random elements in $\mathbb{G}_2$, denoted by $V_i = (v_{i,0}, v_{i,1}, \dots, v_{i,n_i})$. We let $u_{i,j} \leftarrow \psi(v_{i,j})$ and let U_i denote the image of the set V_i under the isomorphism ψ, i.e. $U_i = (u_{i,0}, u_{i,1}, \dots, u_{i,n_i})$. For a bit string B of length n_i we use these sets to define the so-called Waters' hash functions

$$H_i(B) \leftarrow v_{i,0} \prod_{j \in B} v_{i,j},$$

where the product is computed over all values of j for which the j-th bit of B is one. To simplify notation we define

$$G_i(B) \leftarrow u_{i,0} \prod_{j \in B} u_{i,j} = \psi(H_i(B)).$$

Note that $G_i(B)$ can be computed either from the set V_i using the isomorphism ψ, or from the set U_i directly. Also note that $v_{i,j} = h^{\kappa_{i,j}}$ and $u_{i,j} = g^{\kappa_{i,j}}$ for some, unknown, values $\kappa_{i,j} \in \mathbb{Z}_p$.

Our ID-based traitor tracing scheme can now be defined via the following algorithms:

Setup $\mathcal{G}(1^k)$ **:** The key distribution centre generates a set of pairing groups $\mathbb{G}_1, \mathbb{G}_2$ as above at the security level k, along with the sets $V_i \overset{\$}{\leftarrow} (\mathbb{G}_2^*)^{n_i}$ for $i = 1, 2$. A random value $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p$ is selected, and one sets $g_1 \leftarrow g^\alpha \in \mathbb{G}_1$ and $h_1 \leftarrow h^\alpha \in \mathbb{G}_2$. We require a second random element $h_2 \overset{\$}{\leftarrow} \mathbb{G}_2^*$ and we let $g_2 \leftarrow \psi(h_2)$. Finally, the secret random permutation $\pi \overset{\$}{\leftarrow} \mathit{Perm}(N)$ and the secret randomness $r \overset{\$}{\leftarrow} \{0,1\}^\rho$ for the code $\mathbb{C}$ is chosen. The master public key is defined to be $\mathit{mpk} = (g, g_1, h_2, U_1, U_2)$ and the master secret key is $\mathit{msk} = (h, h_2^\alpha, V_1, V_2, \pi, r)$.

Key Extraction $\mathcal{X}(\mathit{msk}, \mathit{ID}, i)$ **:** Let id be the codeword corresponding to index i, i.e. the bit string of length $n_2 = \lceil \log_2 s \rceil \cdot \ell$ that is the binary encoding of codeword $w_r^{(\pi(i))}$. The key distribution centre first select random values $r_1, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$ and then define the private key as

$$d_{\mathit{ID},i} = (\mathit{id}, a_0, a_1, a_2) \leftarrow (\mathit{id}\ ,\ h_2^\alpha H_1(\mathit{ID})^{r_1} H_2(\mathit{id})^{r_2}\ ,\ h^{r_1}\ ,\ h^{r_2})$$

Encryption $\mathcal{E}(\mathit{mpk}, \mathit{ID}, \mathfrak{m})$ **:** A message is defined as an element in $\mathbb{G}_\mathrm{T}$. The sender first chooses a $t \overset{\$}{\leftarrow} \mathbb{Z}_p$ and then computes the ciphertext $C = (C_1, C_2, C_3, C_4) \in \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_T \times \mathbb{G}_1^{n_2+1}$ as

$$C_1 \leftarrow g^t\ , \quad C_2 \leftarrow G_1(\mathit{ID})^t\ , \quad C_3 \leftarrow \mathfrak{m} \cdot \hat{e}(g_1, h_2)^t\ , \quad C_4 \leftarrow (u_{2,j}^t)_{j=0,\dots,n_2}\ .$$

Decryption $\mathcal{D}(d_{ID,i}, C)$ **:** Decryption works as follows, on input of C we first compute

$$C_2' \leftarrow C_4^{(0)} \cdot \prod_{j \in id} C_4^{(j)} = G_2(id)^t \ ,$$

where the last equality follows since $C_4^{(j)} = u_{2,j}^t$. Then we compute

$$\begin{aligned}
C_3 \cdot \frac{\hat{e}(C_2, a_1) \cdot \hat{e}(C_2', a_2)}{\hat{e}(C_1, a_0)} &= \mathfrak{m} \cdot \hat{e}(g_1, h_2)^t \cdot \frac{\hat{e}(G_1(ID)^t, h^{r_1}) \cdot \hat{e}(G_2(id)^t, h^{r_2})}{\hat{e}(g^t, h_2^\alpha H_1(ID)^{r_1} H_2(id)^{r_2})} \\
&= \mathfrak{m} \cdot \hat{e}(g_1, h_2)^t \cdot \frac{\hat{e}(G_1(ID)^{r_1}, h^t) \cdot \hat{e}(G_2(id)^{r_2}, h^t)}{\hat{e}(g^t, h_2^\alpha) \cdot \hat{e}(g^t, H_1(ID)^{r_1} H_2(id)^{r_2})} \\
&= \mathfrak{m} \cdot \frac{\hat{e}(g^\alpha, h_2)^t}{\hat{e}(g^t, h_2^\alpha)} \cdot \frac{\hat{e}\,(g^\sigma, h^t)}{\hat{e}\,(g^t, h^\sigma)} \\
&\quad \text{where } \sigma = r_1(\kappa_{1,0} + \sum_{j \in ID} \kappa_{1,j}) + r_2(\kappa_{2,0} + \sum_{j \in id} \kappa_{2,j}) \\
&= \mathfrak{m} \cdot \frac{\hat{e}(g^\alpha, h_2)^t}{\hat{e}(g^t, h_2^\alpha)} \\
&= \mathfrak{m}.
\end{aligned}$$

Traitor Tracing Algorithm $\mathcal{T}^{\mathbb{D}}(msk, ID)$ **:** Since we use a collusion-secure code, the tracing step requires the secret randomness r, so tracing can only be done by the key distribution centre. The tracing algorithm has access to a pirate box $\mathbb{D}$ that correctly decrypts ciphertexts for ID with probability $\delta(k)$. For convenience, we let $C_4^{(i,j)}$ denote the $(\lceil \log_2 s \rceil (i-1) + j)$-th element of C_4. For each $1 \le i \le \ell$ and $1 \le j \le \lceil \log_2 s \rceil$, initialise counter $ctr_{i,j} \leftarrow 0$ and run the following test $n = 16k/\delta(k)$ times:

1. Choose a random message $\mathfrak{m}$.
2. Encrypt $\mathfrak{m}$ under the group identity ID to form a ciphertext

$$C \leftarrow (C_1, C_2, C_3, C_4).$$

3. Replace $C_4^{(i,j)}$ with a random element from $\mathbb{G}_1$.
4. Query the pirate decoder $\mathbb{D}$ on the altered ciphertext C.
5. If the decoder outputs the message $\mathfrak{m}$ (or a valid ciphertext in the case of minimal access) then increase $ctr_{i,j}$.

After these iterations, reconstruct the bit string id' of length n_2 as follows. Let $id'_{i,j}$ denote the bit of id' at position $\lceil \log_2 s \rceil (i-1)+j$. Set $id'_{i,j} \leftarrow 1$ if $ctr_{i,j} < 4k$, or set $id'_{i,j} \leftarrow 0$ otherwise. Next, decode the bit string id' as a symbol string x of length ℓ, choosing any symbol if the corresponding bit string is not a valid encoding of a symbol in Σ. Finally, use the tracing algorithm of the code to compute $S \xleftarrow{\$} \mathcal{T}_{\mathbb{C}}(x, r)$ and return the set of traitors $\pi^{-1}(S)$.

5 Security Results

The IND-ID-CPA security of our scheme under the DBDH assumption follows from the security of the Waters' HIBE from [17] and an analogue of Theorem 6 of [1]. As one notices that the scheme is simply the Waters WIBE from [1] specialised to the 2-level case. In Appendix A we outline the asymmetric version of Waters' HIBE scheme that we are using.

The scheme as it stands is only secure against adversaries who do not make decryption oracle queries. However, extending to chosen-ciphertext security can be done using the techniques described in [1] based on the techniques of Canetti, Halevi and Katz [9]. This extension will not affect our traitor tracing algorithm given above.

We now turn to showing that our tracing algorithm works. Intuitively, for the $\mathcal{T}_{\mathbb{C}}$ algorithm to work (with error probability ϵ), we need the reconstructed symbol string x to fall within the feasible set of the codewords corresponding to the collusion. This means that on those positions where all the codewords in the collusion are the same, the symbols of x have to be the same as well. We prove that if the ciphertext component $C_4^{(i,j)}$ that is being "tampered" with corresponds to a bit position where all traitors' codewords have a zero, then the pirate box decrypts correctly, unless it can solve the DDH problem in $\mathbb{G}_1$. We also prove that if the tampered component corresponds to an all-one position, then the pirate box is unable to decrypt correctly, unless it can solve the CBDH problem. The $8k/\delta(k)$ iterations are needed because the pirate box only decrypts correctly with probability $\delta(k)$; we use a Chernoff bound to analyse the overall success probability of our tracing algorithm.

Theorem 1. *The $\mathcal{IBTT}$ scheme described above is c-TRA-ID-CPA secure under the assumptions that the underlying code is (c, N, ϵ) collusion-secure code of length ℓ over an alphabet of size s, that the DDH problem in $\mathbb{G}_1$ is hard, and that the CBDH problem in $\mathbb{G}_2$ is hard. More specifically, the advantage of any polynomial-time adversary $\mathcal{A}$ in building an untraceable decoder that correctly decrypts a fraction $\delta(k)$ of ciphertexts using the keys of a collusion of at most c users is at most*

$$\mathbf{Adv}^{\text{tra-id-cpa[c]}}_{\mathcal{A},\mathcal{IBTT}}(k) \;\leq\; \epsilon + \ell\lceil\log_2 s\rceil \cdot \left(\mathbf{Adv}^{\text{cbdh}}_{\mathcal{B}_2,\mathbb{G}_2}(k) + e^{-k}\right)$$

whenever $\delta(k) \geq 2 \cdot \mathbf{Adv}^{\text{xddh}}_{\mathcal{B}_1,\mathbb{G}_1}(k)$ where $\mathcal{B}_1, \mathcal{B}_2$ are polynomial-time algorithms depending on $\mathcal{A}$ and e is the base of the natural logarithm.

Proof. Let $\mathcal{A}$ be an attacker against the tracing property of the encryption scheme; i.e. $\mathcal{A}$ takes as input *mpk* and outputs a pirate decryption box $\mathbb{D}$. We use $\mathcal{A}$ to define an attacker $\mathcal{A}'$ against the tracing property of the collusion-secure code; i.e. $\mathcal{A}'$ will take as input a collection of c random codewords $W = \{w_1, \ldots, w_c\}$ and output a value x. We will prove that if $\mathcal{A}$ successfully avoids being traced, then, with high probability, $\mathcal{A}'$ will successfully output a codeword x that cannot be traced. This will provide the required contradiction.

$\mathcal{A}'$ runs as follows. It chooses random unique indices $i_1, \dots, i_c \in \{1, \dots, N\}$ and mounts the following attack for the collusion $C = \{i_1, \dots, i_c\}$. On input codewords $W = \{w_r^{(i_j)} : j = 1, \dots, c\}$, it first generates a public key $mpk \leftarrow (g, g_1, h_2, U_1, U_2)$ as described in the setup algorithm $\mathcal{G}$ of the identity-based traitor tracing scheme. $\mathcal{A}'$ then runs $\mathcal{A}$. $\mathcal{A}$ may query a key extraction oracle for identities (ID, i) for at most c values of i. $\mathcal{A}'$ responds to the j-th such query as normal using the codeword $w_r^{(i_j)}$. Since W contains codewords corresponding to a random collusion C, and π is meant to be a random permutation, this response is identically distributed to the response of a correct key extraction algorithm. $\mathcal{A}$ terminates by outputting a pirate decryption box $\mathbb{D}$.

$\mathcal{A}'$ then applies the identity-based traitor tracing scheme's tracing algorithm $\mathcal{T}^{\mathbb{D}}$ to $\mathbb{D}$, halting after $\mathcal{T}^{\mathbb{D}}$ determines the value of the symbol string x. $\mathcal{A}'$ outputs the value x. We prove that the symbol string $x \in \Sigma^\ell$ reconstructed by our tracing algorithm falls outside the feasible set $\mathrm{FS}(W)$ with probability at most

$$\Pr\left[x \notin \mathrm{FS}(W)\right] \;\leq\; \ell \lceil \log_2 s \rceil \cdot \left(\mathbf{Adv}^{\mathrm{cbdh}}_{\mathcal{B}_2, \mathbb{G}_2}(k) + e^{-k}\right) \; .$$

The theorem statement then directly follows from the properties of the (c, N, ϵ) collusion-secure code's tracing algorithm $\mathcal{T}_{\mathbb{C}}$.

Let $I \subseteq \{1, \dots, \ell\}$ be the maximal set of symbol positions such that $w_r^{(i)}|_I = w_r^{(j)}|_I$ for all $i, j \in C$. For positions of x not in I there is nothing to prove, because they do not affect membership of $\mathrm{FS}(W)$. So we focus on the symbols x_i of x at positions $i \in I$. Let $id_{i,j}$ for $i \in I$ and $1 \leq j \leq \lceil \log_2 s \rceil$ be the bits in the binary representation of codewords corresponding to symbols at positions $i \in I$. Because of the way we defined I, these bits are the same for all users in the coalition. For a single iteration in the tracing algorithm at position (i, j), the following lemmas upper-bound the probability that the decryption box correctly decrypts $\mathfrak{m}$ in case $id_{i,j} = 0$ and that it does not correctly decrypt $\mathfrak{m}$ in case $id_{i,j} = 1$. Hence, we can distinguish between bit positions which are all zeros and all ones. This means we can recover the symbols which are the same in all the codewords for which the attacker has the keys. If the bits in a given bit position are different in the attacker's codewords, then the attacker can detect the tracing attempt and may output whatever they like. However, this does not matter as we only need to recover the symbols which are the same for all codewords in order to apply the code's tracing algorithm. We postpone the proofs of these lemmas until after the proof of the theorem.

Lemma 1. *If $id_{i,j} = 0$ in the codewords of all users in the collusion C, then $\mathbb{D}$ correctly decrypts a random ciphertext that has been tampered with at position (i, j) with probability*

$$p_0 \;\geq\; \delta(k) - \mathbf{Adv}^{\mathrm{xddh}}_{\mathcal{B}_1, \mathbb{G}_1}(k) \; .$$

Lemma 2. *If $id_{i,j} = 1$ in the codewords of all users in the collusion C, then $\mathbb{D}$ correctly decrypts a random ciphertext that has been tampered with at position (i, j) with probability*

$$p_1 \;\leq\; \mathbf{Adv}^{\mathrm{cbdh}}_{\mathcal{B}_2, \mathbb{G}_2}(k) \; .$$

We also use the following adaptation of the Chernoff bound from [12].

Lemma 3. *Let $X_1, \ldots, X_n$ be independent, 0/1 valued random variables with expected value p. Let $X = X_1 + \ldots + X_n$, let $\mu = \mathbf{E}[X] = np$ and let $0 \le \alpha \le 1$ be a real number. Then we have*

$$\Pr\left[X < (1-\alpha)\mu\right] \ < \ e^{-\mu\alpha^2/2} \ .$$

We want to upper-bound the probability that $x_i \neq w_i$. For a position i, j where $id_{i,j} = 0$, we can see the final value of $ctr_{i,j}$ as the outcome of the sum of $n = 16k/\delta(k)$ independent 0/1 random variables with expected value $p = p_0$. The expected value of $ctr_{i,j}$ is $\mu = np_0$. From Lemma 1 and the assumption that $\mathbf{Adv}^{\text{xddh}}_{\mathcal{B}_1,\mathbb{G}_1}(k) \le \delta(k)/2$, we know that

$$\mu = np_0 \ \ge \ n\big(\delta(k) - \mathbf{Adv}^{\text{xddh}}_{\mathcal{B}_1,\mathbb{G}_1}(k)\big) \ \ge \ \frac{n\delta(k)}{2} = 8k \ .$$

We can then apply the Chernoff bound of Lemma 3 with $\alpha = 1/2$ to upper-bound the probability that the tracing algorithm incorrectly decides that $id'_{i,j} = 1$ by

$$\begin{aligned} \Pr\left[ctr_{i,j} < 4k\right] \ &\le \ \Pr\left[ctr_{i,j} < \mu/2\right] \\ &< \ e^{-\mu/8} \\ &\le \ e^{-k} \ . \end{aligned}$$

On the other hand, for a position i, j where $id_{i,j} = 1$, the probability that the tracing algorithm incorrectly decides that $id'_{i,j} = 0$ can be upper-bounded by

$$\Pr\left[ctr_{i,j} \ge 4k\right] \ \le \ \Pr\left[ctr_{i,j} \ge 1\right] \ = \ p_1 \ \le \ \mathbf{Adv}^{\text{cbdh}}_{\mathcal{B}_2,\mathbb{G}_2}(k) \ .$$

The probability that $x_i \neq w_i$ is upper-bounded by the probability that the tracing algorithm makes an incorrect decision at any of the bit positions. Since there are $\lceil \log_2 s \rceil$ bits in the encoding of x_i, we have that

$$\Pr\left[x_i \neq w_i\right] \ \le \ \lceil \log_2 s \rceil \cdot \left(\mathbf{Adv}^{\text{cbdh}}_{\mathcal{B}_2,\mathbb{G}_2}(k) + e^{-k}\right) \ ,$$

so that the overall probability that the symbol string x reconstructed by the tracing algorithm is not within the feasible set of W is

$$\Pr\left[x \notin \text{FS}(W)\right] \ \le \ \ell\lceil \log_2 s \rceil \cdot \left(\mathbf{Adv}^{\text{cbdh}}_{\mathcal{B}_2,\mathbb{G}_2}(k) + e^{-k}\right) \ ,$$

from which the theorem follows. □

We have left to prove the two lemmas that we used above.

Proof (Lemma 1). For the sake of contradiction, let $\mathcal{A}$ denote an adversary against the traitor tracing scheme that produces a decryption box that correctly decrypts random ciphertexts with probability $\delta(k)$, but that correctly decrypts ciphertexts that have been tampered with at position (i', j') with probability

$p_0 \leq \delta(k) - \gamma$ for some $\gamma > 0$. We will construct an algorithm $\mathcal{B}_1$ which uses $\mathcal{A}$ to gain an advantage γ in solving the DDH problem in $\mathbb{G}_1$.

Let (g^x, g^y, Z) denote the input to our DDH algorithm $\mathcal{B}_1$ and let $k' = s(i'-1)+j'-1$. It constructs the master public keys of the ID-based by choosing random exponents $\alpha, \kappa_{i,j} \xleftarrow{\$} \mathbb{Z}_p^*$ for $a = 1,2$ and $b = 0,\ldots,n_a$ and a random element $h_2 \xleftarrow{\$} \mathbb{G}_2^*$. It sets $g_1 \leftarrow g^\alpha$, $h_1 \leftarrow h^\alpha$, $u_{i,j} \leftarrow g^{\kappa_{i,j}}$, $v_{i,j} \leftarrow h^{\kappa_{i,j}}$, except for $u_{2,k'}$ and $v_{2,k'}$ which it sets to $u_{2,k'} \leftarrow g^x$ and $v_{2,k'} \leftarrow \perp$, respectively. It also chooses secret randomness $r \xleftarrow{\$} \{0,1\}^\rho$ for the collusion-secure code.

$\mathcal{B}_1$ runs $\mathcal{A}$ on input $mpk = (g, g_1, h_2, U_1 = (u_{1,0},\ldots,u_{1,n_1}), U_2 = (u_{2,0},\ldots, u_{2,n_2}))$, responding to its key extraction queries (ID, i) as follows. Let id be the encoding of the codeword $w_r^{(i)}$. We know from the preconditions of the lemma that $id_{k'} = 0$. $\mathcal{B}_1$ chooses $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes the secret key $d_{ID,i} = (id, a_0, a_1, a_2) = (id,\ h_2^\alpha H_1(ID)^{r_1} H_2(id)^{r_2},\ h^{r_1},\ h^{r_2})$. Note that because $id_{k'} = 0$, $\mathcal{B}_1$ can compute $H_2(id)$, even though it does not know $v_{2,k'}$.

At the end of this stage $\mathcal{A}$ will output a pirate decoder $\mathbb{D}$ with respect to a group identity ID of its choice.

All the identities used to create the box $\mathbb{D}$ will have the k'-th bit of their binary code word id set to zero. Algorithm $\mathcal{B}$ then generates a random message $\mathfrak{m}$ and forms the ciphertext

$$C_1 \leftarrow g^y\ , \qquad C_2 \leftarrow (g^y)^{\kappa_{1,0}} \cdot \textstyle\prod_{i \in ID} (g^y)^{\kappa_{1,i}}\ ,$$
$$C_3 \leftarrow \mathfrak{m} \cdot \hat{e}(g^y, h_2)^\alpha\ , \qquad C_4^{(i)} \leftarrow \begin{cases} (g^y)^{\kappa_{2,i}} & \text{for } 0 \leq i \leq n_2,\ i \neq k'\ , \\ Z & \text{for } i = k'\ . \end{cases}$$

This ciphertext is then passed to the decoder $\mathbb{D}$. Algorithm $\mathcal{B}_1$ outputs 1 if the decoder correctly decrypts $\mathfrak{m}$, or outputs 0 otherwise.

If $Z = g^{xy}$, then the ciphertext C is a correctly-formed random ciphertext, so $\mathbb{D}$ will correctly decrypt it with probability $\delta(k)$. If Z is random, then C looks exactly like a ciphertext that has been tampered with at position (i', j'), so $\mathbb{D}$ will correctly decrypt it with probability at most $\delta(k) - \gamma$. The advantage of an algorithm in solving the DDH problem is defined as the difference of the probability that it outputs 1 if $Z = g^{xy}$ and if Z is random, so for our algorithm $\mathcal{B}_1$ we have that

$$\mathbf{Adv}^{\mathrm{xddh}}_{\mathcal{B}_1,\mathbb{G}_1}(k) \ \geq\ \delta(k) - (\delta(k) - \gamma) \ =\ \gamma\ ,$$

from which the lemma follows. □

Proof (Lemma 2). For the sake of contradiction, let $\mathcal{A}$ denote an adversary against the traitor tracing scheme that will produce a decryption box $\mathbb{D}$ that correctly decrypts ciphertexts that have been tampered with at position (i', j') with probability p_1. We will construct an algorithm $\mathcal{B}_2$ which uses $\mathcal{A}$ as a subroutine to solve the bilinear computational Diffie–Hellman problem.

Let h^x, h^y, h^z, be $\mathcal{B}_2$'s input for the CBDH problem. Algorithm $\mathcal{B}_2$ chooses random integers $\kappa_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1,2$ and $0 \leq j \leq n_i$. Let $k' = s(i'-1)+j'-1$.

It sets

$$g_1 \leftarrow \psi(h^x) \qquad h_2 = h^z$$
$$v_{i,j} \leftarrow h^{\kappa_{i,j}} \text{ and } u_{i,j} \leftarrow g^{\kappa_{i,j}} \text{ for } i = 1,2 \text{ and } 0 \le j \le n_i$$

except for $u_{2,k'}$ and $v_{2,k'}$ which it sets to

$$v_{2,k'} \leftarrow h^{\kappa_{2,k'}}/h^x = h^{\kappa_{2,k'}-x} \qquad u_{2,k'} \leftarrow \psi(v_{2,k'}) = g^{\kappa_{2,k'}-x} .$$

It also chooses secret randomness $r \xleftarrow{\$} \{0,1\}^\rho$ for the collusion-secure code. It then runs $\mathcal{A}$ on input $mpk = (g, g_1, h_2, (u_{1,0}, \ldots, u_{1,n_1}), (u_{2,0}, \ldots, u_{2,n_2}))$.

Algorithm $\mathcal{A}$ will make c key extraction queries (ID, i). Let id be the codeword corresponding to user i; we know from the preconditions of the lemma that $id_{k'} = 1$ for all users in the collusion. The decryption key $d_{ID,i} = (id, a_0, a_1, a_2)$ is generated by choosing $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ at random and computing

$$\begin{aligned}
a_0 &\leftarrow (h^z)^{\kappa_{2,k'}} \cdot (h^x)^{-r_2} \cdot h^{\kappa_{2,k'}r_2} \cdot H_1(ID)^{r_1} \cdot (h^z \cdot h^{r_2})^{\kappa_{2,0}} \cdot \prod_{i \in id, i \neq k'} (h^z \cdot h^{r_2})^{\kappa_{2,i}} \\
&= h^{z\kappa_{2,k'} - xr_2 + \kappa_{2,k'}r_2} \cdot H_1(ID)^{r_1} \cdot \left(h^{\kappa_{2,0}} \prod_{i \in id, i \neq k'} h^{\kappa_{2,i}} \right)^{z+r_2} \\
&= h^{xz - xz + z\kappa_{2,k'} - xr_2 + \kappa_{2,k'}r_2} \cdot H_1(ID)^{r_1} \cdot \left(v_{2,0} \prod_{i \in id, i \neq k'} v_{2,i} \right)^{z+r_2} \\
&= h^{xz} \cdot H_1(ID)^{r_1} \cdot H_2(id)^{z+r_2} \\
a_1 &\leftarrow h^{r_1}, \\
a_2 &\leftarrow h^z \cdot h^{r_2} \;=\; h^{z+r_2}
\end{aligned}$$

At the end of this stage $\mathcal{A}$ will output a pirate decoder $\mathbb{D}$ with respect to a group identity ID of its choice. Algorithm $\mathcal{B}_2$ then generates the challenge ciphertext with

$$\begin{aligned}
&C_1 \leftarrow \psi(h^y) \;, \quad C_2 \leftarrow \psi(h^y)^{\kappa_{1,0}} \prod_{i \in ID} \psi(h^y)^{\kappa_{1,i}} \;, \\
&C_3 \xleftarrow{\$} \mathbb{G}_\mathrm{T} \;, \quad C_4^{(i)} \leftarrow \begin{cases} \psi(h^y)^{\kappa_{2,i}} & \text{for } 0 \le i \le n_2,\ i \neq k' \;, \\ Z & \text{where } Z \xleftarrow{\$} \mathbb{G}_1 \text{ for } i = k' \;. \end{cases}
\end{aligned}$$

By our assumption on the pirate decoder $\mathbb{D}$ with this ciphertext will output, with probability p_1, the corresponding plaintext $\mathfrak{m}$ as if $C_4^{(k')}$ were chosen correctly as $u_{2,k'}^y$. In this case $\mathcal{B}_2$ can recover $\hat{e}(g,h)^{xyz}$ by computing $C_3/\mathfrak{m}$. Algorithm $\mathcal{B}_2$ then returns this value as its solution to the bilinear computational Diffie–Hellman problem, giving it an advantage

$$\mathbf{Adv}^{\mathrm{cbdh}}_{\mathcal{B}_2,\mathbb{G}_2}(k) \;\ge\; p_1 \;,$$

from which the lemma follows. □

Acknowledgements. We would like to thank Yevgeniy Dodis and Aggelos Kiayias for suggesting that a simple method for the converting the q-ary alphabet into binary is sufficient for our purposes.

References

1. M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, and N. Smart. Identity-based encryption gone wild. In *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 300–311. Springer-Verlag, 2006.
2. L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive, Report 2005/417, 2005. `http://eprint.iacr.org/`.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, 1993.
4. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, 2004.
5. D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer-Verlag, 1999.
6. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.
7. D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer-Verlag, 2006.
8. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer-Verlag, 1995.
9. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer-Verlag, 2004.
10. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363. Springer-Verlag, 2001.
11. A. Kiayias and M. Yung. On crafty pirates and foxy tracers. In *ACM CCS Digital Rights Management Workshop 2001*, volume 2320 of *LNCS*, pages 22–39. Springer-Verlag, 2002.
12. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
13. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, 2000.
14. M. Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/`.
15. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1985.
16. G. Tardos. Optimal probabilistic fingerprint codes. In *35th ACM STOC*, pages 116–125. ACM Press, 2003.
17. B. R. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer-Verlag, 2005.

A Waters' HIBE with Asymmetric Pairings

Our scheme is built out of the HIBE suggested by Waters in [17], but in the asymmetric pairing setting and using a scheme of depth 2. In this section we describe the underlying HIBE in full generality.

A.1 Scheme Description

Suppose that we want a scheme of depth L. We define L sets $V_1, \ldots, V_L$ of random elements in $\mathbb{G}_2$, with elements denoted $V_i = (v_{i,0}, v_{i,1}, \ldots, v_{i,n_i})$. We let $u_{i,j} = \psi(v_{i,j})$ and let U_i denote the image of the set V_i under the isomorphism ψ, i.e. $U_i = (u_{i,0}, u_{i,1}, \ldots, u_{i,n_i})$.

Just as in our traitor tracing scheme for a bit string B of length n_i we use these sets to define the Waters' hash functions:

$$H_i(B) = v_{i,0} \prod_{j \in B} v_{i,j},$$

where the products are over all the set bits in B. To simplify notation we define

$$G_i(B) = u_{i,0} \prod_{j \in B} u_{i,j} = \psi(H_i(B)).$$

Note that $\psi(H_i(B)) = G_i(B)$ can be computed either from the set V_i using the isomorphism ψ, or from the set U_i directly. Also note that $v_{i,j} = h^{\kappa_{i,j}}$ and $u_{i,j} = g^{\kappa_{i,j}}$ for some, unknown, values $\kappa_{i,j} \in \mathbb{Z}_p$.

Using the entities above, the various algorithms that make up Waters' HIBE scheme are as follows. We assume that id is a tuple $(id_1, \ldots, id_l)$ where $l \leq L$ and id_i is a bit string of length n_i, applying a collision resistant hash function if necessary.

Setup $\mathcal{G}(1^k)$ **:** We generate a set of pairing groups as above at the security level k, along with the sets $V_1, \ldots, V_L$ and $U_1, \ldots, U_L$. We require a random element $h \xleftarrow{\$} \mathbb{G}_2$ and let $g \leftarrow \psi(h) \in \mathbb{G}_1$. A random value $\alpha \xleftarrow{\$} \mathbb{Z}_p$ is selected, and we set $g_1 \leftarrow g^\alpha$ and $h_1 \leftarrow h^\alpha$. We require a second random element $h_2 \in \mathbb{G}_2$ and we let $g_2 \leftarrow \psi(h_2)$. The master public key is defined to be $mpk = \{g, g_1, h_2, U_1, \ldots, U_L\}$ and the master secret key is $msk = \{h, h_2^\alpha, V_1, \ldots, V_L\}$.

Key Extraction $\mathcal{X}(id, msk)$ **:** We first select random values $r_1, \ldots, r_l \leftarrow \mathbb{Z}_p$ and then define the private key as

$$d_{id} = (a_0, a_1, \ldots, a_l) \leftarrow \left(h_2^\alpha \prod_{i=1}^{l} H_i(id_i)^{r_i}, h^{r_1}, \ldots, h^{r_l} \right) \in \mathbb{G}_2^{l+1}.$$

Encryption $\mathcal{E}(id, mpk, \mathfrak{m})$ **:** A message is defined as an element in $\mathbb{G}_T$. The sender first choose a $t \leftarrow \mathbb{Z}_p$ and then computes the ciphertext

$$C = (C_1, C_2, C_3) \in \mathbb{G}_1 \times \mathbb{G}_1^l \times \mathbb{G}_\mathrm{T}$$

as

$$C_1 \leftarrow g^t, \quad C_2 \leftarrow \left(C_{2,i} = G_i(id_i)^t\right)_{i=1}^{l}, \quad C_3 \leftarrow \mathfrak{m} \cdot \hat{e}(g_1, h_2)^t.$$

Decryption $\mathcal{D}(C, d_{id})$ **:** Compute

$$C_3 \cdot \frac{\prod_{i=1}^{l} \hat{e}(C_{2,i}, a_i)}{\hat{e}(C_1, a_0)} = \mathfrak{m}.$$