Tracing Malicious Proxies in Proxy Re-Encryption

Benoît Libert[1] and Damien Vergnaud[2]

[1] Université Catholique de Louvain, Crypto Group
Place du Levant, 3 – 1348 Louvain-la-Neuve, Belgium
[2] Ecole Normale Supérieure – C.N.R.S. – I.N.R.I.A.
45, Rue d'Ulm – 75230 Paris CEDEX 05 – France

Abstract. In 1998, Blaze, Bleumer and Strauss put forth a cryptographic primitive, termed *proxy re-encryption*, where a semi-trusted proxy is given some piece of information that enables the re-encryption of ciphertexts from one key to another. Unidirectional schemes only allow translating from the delegator to the delegatee and not in the opposite direction. In all constructions described so far, although colluding proxies and delegatees cannot expose the delegator's long term secret, they can derive and disclose sub-keys that suffice to open all translatable ciphertexts sent to the delegator. They can also generate new re-encryption keys for receivers that are not trusted by the delegator. In this paper, we propose *traceable proxy re-encryption* systems, where proxies that leak their re-encryption key can be identified by the delegator. The primitive does not preclude illegal transfers of delegation but rather strives to deter them. We give security definitions for this new primitive and a construction meeting the formalized requirements. This construction is fairly efficient, with ciphertexts that have logarithmic size in the number of delegations, but uses a non-black-box tracing algorithm. We discuss how to provide the scheme with a black box tracing mechanism at the expense of longer ciphertexts.

Keywords. unidirectional proxy re-encryption, transferability issues, collusion detection and traceability.

1 Introduction

Ten years ago, Blaze, Bleumer and Strauss proposed a cryptographic primitive called *proxy re-encryption* (PRE), in which a proxy transforms – without being able to infer any information on the corresponding plaintext – a ciphertext computed under Alice's public key into one that can be opened using Bob's secret key. In all known constructions, if Bob and a malicious proxy cooperate, they can derive new re-encryption keys without Alice's consent. The purpose of this paper is to coin a new notion, that we call *traceable proxy re-encryption* (TPRE) in which such misbehaving proxies can be identified by the delegator. We formalize security notions for this new primitive and give a reasonably efficient construction fitting this model under appropriate complexity assumptions.

Related work. Blaze *et al.* [8] proposed the first PRE scheme, where plaintexts and secret keys remain hidden from the proxy. Unfortunately, their scheme has inherent limitations: the proxy key also allows translating ciphertexts from Bob to Alice, which may be undesirable, and the proxy and the delegatee can collude to expose the delegator's private key.

In 2005, Ateniese, Fu, Green and Hohenberger [4, 5] showed how to construct *unidirectional* schemes using bilinear maps and simultaneously prevent proxies from colluding with delegatees in order to expose the delegator's long term secret. Their schemes involve two distinct encryption algorithms: *first-level* encryptions are not translatable whilst *second-level* encryptions can be re-encrypted by proxies into ciphertexts that are openable by delegatees. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi)$ be a cryptographic bilinear structure (denoted multiplicatively) of prime order p and let g be a generator of $\mathbb{G}_1$ (see § 2.2 for a definition). Alice and Bob publish the public keys $y_A = g^a$ and $y_B = g^b$ (respectively) and keep secret their discrete logarithms a and b. To encrypt a message $m \in \mathbb{G}_T$ to Alice at the second level, a sender picks a random $r \in \mathbb{Z}_p^*$ and transmits the pair (c_1, c_2) where $c_1 = y_A^r$ and $c_2 = m \cdot e(g, h)^r$ where $h = \psi(g)$. The proxy is given the re-encryption key $h^{b/a}$ and can translate ciphertexts from Alice to Bob by computing $(e(c_1, h^{b/a}), c_2) = (e(g, h)^{br}, m \cdot e(g, h)^r)$. The decryption operations are somewhat similar to those of the Elgamal [18] cryptosystem. This strategy does not completely withstand collusions since, if Bob and the proxy cooperates, they obtain the element $h^{1/a}$ which suffices to decrypt any second-level ciphertext intended to Alice. Even if the last few years saw a renewed interest in proxy re-encryption [4, 5, 25, 16, 19, 24], all known constructions entail to trust proxies not to collude with certain participants. Otherwise, sub-keys such as $h^{1/a}$ or new re-encryption keys can be derived and disclosed over the Internet.

Transferability issues in proxy re-encryption. Following [21], a PRE scheme is said *non-transferable* if the proxy and a set of colluding delegatees cannot re-delegate their decryption rights. The first question that comes to mind is whether transferability is really preventable since the delegatee can always decrypt and forward the plaintext. However, the difficulty in retransmitting data restricts this behavior. The security goal is therefore to prevent the delegatee and the proxy to provide another party with a secret value that can be used *offline* to decrypt the delegator's ciphertexts. Obviously, the delegatee can always send its secret key to this party, but in doing so, it assumes a security risk that is potentially injurious to itself. In the simple aforementioned unidirectional system, colluders can unfortunately disclose $h^{1/a}$ which is clearly harmless to the cheating delegatee and allows for the offline opening of second level ciphertexts encrypted for the delegator. All other existing unidirectional [5] schemes are actually vulnerable to this kind of attack.

A desirable security goal is therefore to prevent a malicious proxy (or a collusion of several rogue proxies) interacting with users to take such actions. To the best of our knowledge, this non-transferability property has been elusive in the literature. This is not surprising since, given that proxies and delegatees can always decrypt level 2 ciphertexts by combining their secrets, they must be able

to jointly compute data that allows decrypting and, once revealed to a malicious third party, ends up with a transfer of delegation. Therefore, discouraging such behaviors seems much easier than preventing them.

Our contributions. We introduce a new notion, that we call *traceable proxy re-encryption* (TPRE), where proxies that reveal their re-encryption key to third parties can be identified by the delegator. The primitive does not preclude illegal transfers of delegation but provides a disincentive to them. Unlike prior unidirectional PRE systems, when delegators come across an illegally formed re-encryption key, they can determine its source among potentially malicious proxies. It also allows tracing delegatees and proxies that pool their secrets to disclose a pirate decryption sub-key which suffices to decipher ciphertexts originally intended for the delegator. Identifying dishonest delegatees is useful in applications such as PRE-based file storage systems [4, 5] where there is a single proxy (i.e. the access control server) and many delegatees (i.e. end users). When a pirate decryption sub-key is disclosed in such a situation, we can find out which client broke into the access control server to generate it.

Deterring potentially harmful actions from parties that are *a priori* trustworthy may seem overburden: no one would elect a delegatee without having high confidence in his honesty. In these regards, the present work is somehow related to ideas from Goyal [20] that aim at avoiding to place too much trust in entities (i.e. trusted authorities in identity-based encryption schemes) that must be trusted anyway. Arguably, users are less reluctant to grant their trust when abuses of delegated power are detectable and discouraged.

We formalize security notions for TPRE and give efficient implementations meeting these requirements under different pairing-related assumptions. Our constructions borrow techniques from *traitor tracing* schemes [17]. We also make use of a special kind of *identity-based encryption* (IBE) system (where arbitrary strings such as email addresses [27, 11] can act as a public keys so as to avoid costly digital certificates), introduced in 2006 by Abdalla *et al.* and called *wildcard identity-based encryption* (WIBE) [1].

Our main scheme is fairly efficient, with ciphertexts of logarithmic size in the number of delegations, but the tracing system is non-black-box. Its security relies on (formerly used) mild pairing-related assumptions and the security analysis takes place in the standard model (without the random oracle heuristic [7]).

We also discuss how the scheme can be equipped with a black-box tracing mechanism at the expense of longer ciphertexts. The design principle is to associate re-encryption keys with codewords from a collusion-secure code [14]. This scheme is inspired from a WIBE-based identity-based traitor tracing scheme [2] and inherits its disadvantages: its computational overhead and the size of ciphertexts are linear in the length of the underlying code.

Roadmap. In the upcoming sections, we first define the concept of TPRE scheme and its security model. Then, we describe the intractability assumption that our scheme relies on. In section 3, we detail our scheme and first provide some intuition of the underlying idea. We finally give security results. Section 4 briefly explains how to obtain a black-box tracing mechanism.

2 Preliminaries

2.1 Model and security notions

Definition 1. *A (single hop) unidirectional PRE scheme is a tuple of algorithms* $(\mathsf{Global\text{-}setup}, \mathsf{Keygen}, \mathsf{ReKeygen}, \mathsf{CheckKey}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{ReEnc}, \mathsf{Dec}_1, \mathsf{Dec}_2)$*:*

- $\mathsf{Global\text{-}setup}(\lambda) \rightarrow \mathsf{par}$*: on input of a security parameter* λ*, this algorithm produces public parameters* par *to be used by all parties.*
- $\mathsf{Keygen}(\lambda, \mathsf{par}) \rightarrow (sk, pk)$*: on input of common public parameters* par *and a security parameter* λ*, all parties use this randomized algorithm to generate a private/public key pair* (sk, pk)*.*
- $\mathsf{ReKeygen}(\mathsf{par}, sk_i, pk_j) \rightarrow R_{ij}$*: given public parameters* par*, user* i*'s private key* sk_i *and user* j*'s public key* pk_j*, this (possibly randomized) algorithm outputs a key* R_{ij} *that allows re-encrypting second level ciphertexts intended to* i *into first level ciphertexts encrypted for* j*.*
- $\mathsf{CheckKey}(\mathsf{par}, sk_i, pk_j, R_{ij}) \rightarrow b \in \{0,1\}$*: is a deterministic algorithm checking the well-formedness of* R_{ij} *as a proxy key for re-encrypting messages from user* i *to user* j*.*
- $\mathsf{Enc}_1(\mathsf{par}, pk, m) \rightarrow C$*: on input of public parameters* par*, a receiver's public key* pk *and a plaintext* m*, this probabilistic algorithm outputs a first level ciphertext that cannot be re-encrypted for another party.*
- $\mathsf{Enc}_2(\mathsf{par}, pk, m) \rightarrow C$*: given public parameters* par*, a receiver's public key* pk *and a plaintext* m*, this randomized algorithm outputs a second level ciphertext that can be re-encrypted into a first level ciphertext (intended to a possibly different receiver) using the appropriate re-encryption key.*
- $\mathsf{ReEnc}(\mathsf{par}, R_{ij}, C) \rightarrow C'$*: this (possibly randomized) algorithm takes as input public parameters* par*, a re-encryption key* R_{ij} *and a second level ciphertext* C *encrypted for user* i*. The output is a first level ciphertext* C' *re-encrypted for user* j*. In single hop schemes,* C' *cannot be re-encrypted any further.*
- $\mathsf{Dec}_1(\mathsf{par}, sk, C) \rightarrow m$*: given a private key* sk*, a first level ciphertext* C *and system-wide parameters* par*, this algorithm outputs a plaintext* $m \in \{0,1\}^*$*.*
- $\mathsf{Dec}_2(\mathsf{par}, sk, C) \rightarrow m$*: given a private key* sk*, a second level ciphertext* C *and public parameters* par*, this algorithm returns a plaintext* $m \in \{0,1\}^*$*.*

For any common public parameters par*, any message* $m \in \{0,1\}^*$ *and any couple of private/public key pair* (sk_i, pk_i)*,* (sk_j, pk_j) *these algorithms should satisfy the following correctness conditions:*

$$\mathsf{Dec}_1(\mathsf{par}, sk_i, \mathsf{Enc}_1(\mathsf{par}, pk_i, m)) = m;$$
$$\mathsf{Dec}_2(\mathsf{par}, sk_i, \mathsf{Enc}_2(\mathsf{par}, pk_i, m)) = m;$$
$$\mathsf{Dec}_1(\mathsf{par}, sk_j, \mathsf{ReEnc}(\mathsf{par}, \mathsf{ReKeygen}(\mathsf{par}, sk_i, pk_j), \mathsf{Enc}_2(\mathsf{par}, pk_i, m))) = m;$$
$$\mathsf{CheckKey}(\mathsf{par}, sk_i, pk_j, \mathsf{ReEnc}(\mathsf{par}, \mathsf{ReKeygen}(\mathsf{par}, sk_i, pk_j))) = 1.$$

In a traceable PRE scheme, there is an additional procedure Trace which, given user i's private key sk_i as well as a pirate proxy key R_{ij}^{bad} allowing for illegal translations from i to another user j, outputs the identity of at least one of the

malicious proxies that made up R_{ij}^{bad}. Algorithm Trace can also take as input a pirate decryption key $R_{i\star}^{\mathsf{bad}}$ that, instead of re-encrypting second level ciphertexts intended for user i, simply directly recovers the underlying plaintext. In this case, the tracing algorithm should also determine which malicious delegatee has colluded with the incriminated proxy to generate of $R_{i\star}^{\mathsf{bad}}$.

Semantic security. As in [4, 5, 16], we require that users publicize public keys only if they hold the corresponding private keys. This amounts to adopt a trusted key generation model or a model where all parties have to prove knowledge of their secret keys when registering their public keys upon certification.

Like [4, 5, 16], we also assume a static model where adversaries do not choose whom to corrupt depending on the information gathered so far.

Definition 2. *A (single-hop) unidirectional PRE scheme is semantically secure at level* 2 *if the probability*

$$\begin{aligned}\Pr[(pk^\star, sk^\star) \leftarrow \mathsf{Keygen}(\lambda), \{(pk_x, sk_x) \leftarrow \mathsf{Keygen}(\lambda)\}, \{(pk_h, sk_h) \leftarrow \mathsf{Keygen}(\lambda)\},\\ \{R_{x\star} \leftarrow \mathsf{ReKeygen}(sk_x, pk^\star)\},\\ \{R_{\star h} \leftarrow \mathsf{ReKeygen}(sk^\star, pk_h)\}, \{R_{h\star} \leftarrow \mathsf{ReKeygen}(sk_h, pk^\star)\},\\ \{R_{hx} \leftarrow \mathsf{ReKeygen}(sk_h, pk_x)\}, \{R_{xh} \leftarrow \mathsf{ReKeygen}(sk_x, pk_h)\},\\ \{R_{hh'} \leftarrow \mathsf{ReKeygen}(sk_h, pk_{h'})\}, \{R_{xx'} \leftarrow \mathsf{ReKeygen}(sk_x, pk_{x'})\},\\ (m_0, m_1, St) \leftarrow \mathcal{A}\big(pk^\star, \{(pk_x, sk_x)\}, \{pk_h\}, \{R_{x\star}\}, \{R_{h\star}\},\\ \{R_{\star h}\}, \{R_{xh}\}, \{R_{hx}\}, \{R_{hh'}\}, \{R_{xx'}\}\big),\\ d^\star \stackrel{R}{\leftarrow} \{0,1\}, C^\star = \mathsf{Enc}_2(m_{d^\star}, pk^\star), d' \leftarrow \mathcal{A}(C^\star, St) :\\ d' = d^\star]\end{aligned}$$

is negligibly (as a function of the security parameter λ) close to $1/2$ *for any PPT adversary $\mathcal{A}$. In our notation, St is a state information maintained by $\mathcal{A}$ while $(pk^\star, sk^\star)$ is the target user's key pair generated by the challenger that also chooses other keys for corrupt and honest parties. For other honest parties, keys are subscripted by h or h' and we subscript corrupt keys by x or x'. The adversary is granted access to all re-encryption keys but those for re-encrypting from the target user to a corrupt one. $\mathcal{A}$ is said to have advantage ε if this probability, taken over all coin tosses, is at least* $1/2 + \varepsilon$.

Security of first level ciphertexts. Definition 2 provides adversaries with a second level challenge ciphertext. An orthogonal definition captures $\mathcal{A}$'s inability to distinguish first level ciphertexts as well. For *single-hop* schemes, the adversary is allowed to see *all* re-encryption keys in this definition. As first level ciphertexts cannot be re-encrypted any further, there is no reason to hold specific honest-to-corrupt re-encryption keys back from the adversary. A unidirectional scheme fitting this definition is said semantically secure at the first level.

Digital-identity security in PRE. In [4], Ateniese *et al.* define an important security requirement for unidirectional PRE schemes. This notion, termed *master*

secret security or *digital-identity security*, demands that no coalition of dishonest delegatees and proxies be able to pool their keys in order to expose the private key of their delegator. More formally, the following probability should be negligible as a function of the security parameter λ. In our notations, we superscript pk and sk with $\star$ to denote the keys of the target honest user whereas adversarial users' keys are subscripted by x.

$$\begin{aligned}\Pr[\ & (pk^\star, sk^\star) \leftarrow \mathsf{Keygen}(\lambda), \{(pk_x, sk_x) \leftarrow \mathsf{Keygen}(\lambda)\},\\ & \{R_{\star x} \leftarrow \mathsf{ReKeygen}(sk^\star, pk_x)\}, \{R_{x\star} \leftarrow \mathsf{ReKeygen}(sk_x, pk^\star)\},\\ & \gamma \leftarrow A(pk^\star, \{(pk_x, sk_x)\}, \{R_{\star x}\}, \{R_{x\star}\}) : \gamma = sk^\star\ ]\end{aligned}$$

While reasonable in many applications, this definition does not consider colluding delegatees and proxies who attempt to produce a new re-encryption key $R_{\star x'}$ that was not originally given and allows re-encrypting from the target user to another malicious party x'. As already stressed, all known unidirectional PRE schemes fail to resist such attacks. Although colluders are unable to expose the delegator's long term secret $sk^\star$, they can still compute a sub-key sk^{bad} that allows decrypting ciphertexts at level 2. To address this issue, our model asks that the cheated delegator be able to determine – at least partially and with high probability – where the illegal transfer of delegation stems from or who crafted the pirate sub-key sk^{bad}. In our scheme, this unfortunately comes at the expense of sacrificing the key and ciphertext optimality properties met in [4, 5].

Traceability. Consider a set of proxies $P_1, P_2, \ldots, P_N$ that receive re-encryption keys allowing for the translation of ciphertexts from user A to his delegatees $B_1, B_2, \ldots, B_N$. We say that a PRE scheme is *traceable* if any subset of these proxies colluding with delegatees $B_1, B_2, \ldots, B_N$ is unable to generate a new re-encryption key that cannot be traced back to one of them.

Definition 3. *A unidirectional PRE scheme is* ***traceable*** *if no PPT adversary $\mathcal{A}$ has non-negligible probability of success in the following game:*

1. *The challenger provides $\mathcal{A}$ with the target user's public key pk_0, public keys pk_i for other honest parties and key pairs (sk_i, pk_i) for corrupt users.*
2. *On multiple occasions, $\mathcal{A}$ may invoke a re-encryption key generation oracle $\mathcal{O}_{rkey}$. When queried on input of public keys (pk_i, pk_j) that were both obtained from the challenger, this oracle returns the re-encryption key $R_{ij} = \mathsf{ReKeygen}(sk_i, pk_j)$. Let T be the set of proxy keys obtained by $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a pirate re-encryption key $R^\star_{0t}$ together with a public key pk_t that belongs to the public key space of the scheme (i.e. for which an associated private key exists) and differs from public keys of the target user's delegatees. The adversary is declared successful if the following two conditions hold:*
   a. *$\mathsf{CheckKey}(sk_0, pk_t, R^\star_{0t}) = 1$ (i.e. $R^\star_{0t}$ is a valid re-encryption key).*
   b. *The tracing procedure (run by the challenger on $R^\star_{0t}$ using the target user's secret sk_0) fails to identify a correct proxy key $R^{\mathsf{bad}}_{0j} \in T$. That is, if $R^{\mathsf{bad}}_{0j} = \mathsf{Trace}(sk_0, R^\star_{0t}, pk_t)$, we have either $R^{\mathsf{bad}}_{0j} = \perp$ or if $R^{\mathsf{bad}}_{0j} \notin T$.*

The pirate key $R^{\star}_{0t}$ should re-encrypt from user 0 to a user having public key pk_t. For simplicity, we assume that the latter is supplied by $\mathcal{A}$ at the end of the game. When the target user finds a suspicious re-encryption key $R^{\star}$ in practice, he does not *a priori* know to whom ciphertexts can be re-encrypted using $R^{\star}$. However, he can determine it by simply testing whether $\mathsf{CheckKey}(sk_0, \tilde{pk}_j, R^{\star}) = 1$, for $j = 1, \ldots, \eta$, given a set of suspicious public keys $\{\tilde{pk}_1, \ldots, \tilde{pk}_\eta\}$.

We insist that pk_t may differ from public keys that are generated by the challenger at step 1 of the game. Besides, the definition does not force $\mathcal{A}$ to reveal the matching private key sk_t to the challenger: the only requirement is that such a private key exists.

At first glance, one may wonder why $\mathcal{A}$ should be allowed to come up with an arbitrary pk_t of her choosing whilst delegation queries to $\mathcal{O}_{rkey}$ are only permitted for delegatees' public keys that were chosen by the challenger.

We actually find it natural to assume that honest users only delegate to parties whose public key has been properly certified and for which knowledge of the underlying secret key has been demonstrated to the CA at key registration. In contrast, pk_t is not meant to have a legal use and simply provides a way to covertly translate the target user's communications. Hence, there is no reason to assume that the challenger learns sk_t whatsoever. Finally, when the proxy is compromised but the delegatee j remains honest, the adversary obtains R_{0j} such that $\mathsf{CheckKey}(sk_0, pk_j, R_{0j}) = 1$. Then, she might be able to compute $R^{\star}_{0t}$ and pk_t (as a function of pk_j) such that $\mathsf{CheckKey}(sk_0, pk_t, R^{\star}_{0t}) = 1$. In this case, the adversary clearly does not know sk_t. The property that we require is that $R^{\star}_{0t}$ can be traced back to the proxy involved in its creation. Then, if pk_t happens to be a registered public key (for which a proof of knowledge of the underlying private key was provided), the delegator figures out that the original delegatee was also part of the collusion, as well as the user holding sk_t.

Bounded Traceability. Similarly to common situations in traitor tracing schemes, it may happen that traceability is guaranteed only if the adversary makes at most k re-encryption key queries involving the secret sk_0 of the target user acting as a delegator (regardless of whether the delegatee is honest). On the other hand, she is granted as many re-encryption key queries involving other honest delegators as she likes. Schemes that are secure in this scenario are said k-traceable.

Black Box Traceability. A new analogy with traitor tracing primitives suggests to strengthen the definition by assuming that the adversary only outputs a re-encryption device $\mathbb{P}$ that translates ciphertexts with non-negligible probability but cannot be reverse-engineered so as to extract the built-in key. Indeed, it has been reported [22] that proxy re-encryption systems can be safely obfuscated. It would thus be desirable to have a black-box tracing procedure to recover the identity of colluding parties using $\mathbb{P}$ as a re-encryption oracle. A variant of our scheme can be equipped with a limited black-box tracing mechanism. Due to the use of collusion-secure codes [14], this variant unfortunately features unreasonably large ciphertexts and cannot be considered as being practical. Moreover, it only tolerates a bounded number of traitors k. Lastly, it does not allow to

determine who the dishonest delegatees are when running a pirate decryption device $\mathbb{D}$ in tracing mode: only colluding proxies can be traced.

2.2 Bilinear Maps and Complexity Assumptions

We consider a configuration of *bilinear map groups* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p with a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ satisfying the following properties:

1. bilinearity: $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}$;
2. efficient computability for any input pair;
3. non-degeneracy: $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g \neq 1_{\mathbb{G}_1}$ and $h \neq 1_{\mathbb{G}_2}$.

We will need an extension of the Decision Bilinear Diffie-Hellman (DBDH) assumption which is the intractability of distinguishing $e(g, h)^{abc}$ given (h^a, h^b, h^c).

Definition 4. *In bilinear map groups* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$*, The* **Augmented Decision Bilinear Diffie-Hellman Problem** *(ADBDH) is to distinguish* $e(g, h)^{abc}$ *from random elements of* $\mathbb{G}_T$ *given* $(g, h, h^a, h^b, h^c, h^{a^2 b}) \in \mathbb{G}_1 \times \mathbb{G}_2^5$*. A distinguisher* $\mathcal{D}$ (τ, ε)*-breaks the assumption if it has running time* τ *and*

$$Adv(\mathcal{D}) = |\Pr[\mathcal{D}(h^a, h^b, h^c, h^{a^2 b}, e(g, h)^{abc}) = 1 | a, b, c \xleftarrow{R} \mathbb{Z}_p^*] - \Pr[\mathcal{D}(h^a, h^b, h^c, h^{a^2 b}, e(g, h)^z) = 1 | a, b, c, z \xleftarrow{R} \mathbb{Z}_p^*]| \geq \varepsilon$$

This problem is not easier than breaking the ℓ-Bilinear Diffie-Hellman Exponent (ℓ-BDHE) assumption of [12] that implies the infeasibility of recognizing $e(\psi(h'), h)^{(a^{\ell+1})}$ given $(h', h, h^a, h^{(a^2)}, \ldots, h^{(a^\ell)}, h^{(a^{\ell+2})}) \in \mathbb{G}_2^{\ell+4}$. When $a = b$, ADBDH boils down to a special case[3] of ℓ-BDHE with $\ell = 1$. The generic hardness ADBDH is thus implied by that of ℓ-BDHE, which was shown in [10].

Our proof of traceability relies on a problem named *2-out-of-3 Diffie-Hellman* in [23], where its generic intractability was shown in prime order groups. A not harder version of this problem was previously considered in [3].

Definition 5. *The* **2-out-of-3 Diffie-Hellman** *problem (2-3-CDH) is, given* $(h, h^a, h^b) \in \mathbb{G}^3$*, to find a pair* $(C, C^{ab}) \in \mathbb{G} \times \mathbb{G}$ *with* $C \neq 1_{\mathbb{G}}$.

3 A Scheme with Logarithmic Complexity

This section presents our main scheme providing non-black-box traceability. It borrows ideas from the identity-based traitor tracing described in [2].

[3] It is actually the hardness of deciding if $T \stackrel{?}{=} e(g, h)^{a^2 c}$ given $(h' = h^c, h^a, h^{(a^3)})$.

3.1 Intuition

To provide a better intuition of the scheme, we need the recall the Waters IBE [29] and the notion of wildcard IBE [1]. The former involves a trusted party that publishes a master public key $mpk = (Z = e(g,h)^z, V_0, V_1, \ldots, V_n) \in \mathbb{G}_T \times \mathbb{G}_2^{n+1}$ where $z \xleftarrow{R} \mathbb{Z}_p^*$ and n is the length of identity strings. The trusted authority keeps a master secret $msk = h^z$ to itself. This secret is used to derive private keys from user's identities $id = i_1 \ldots i_n \in \{0,1\}^n$ by computing

$$d_{id} = (d_1, d_2) = \big(msk \cdot (V_0 \cdot \prod_{\ell=1}^{n} V_\ell^{i_\ell})^r, h^r\big)$$

for a randomly chosen exponent $r \xleftarrow{R} \mathbb{Z}_p^*$. Such a private key always satisfies

$$e(g, d_1) = Z \cdot e(U_0 \cdot \prod_{\ell=1}^{n} U_\ell, d_2) \tag{1}$$

where $U_\ell = \psi(V_\ell)$ for $\ell = 0, \ldots, n$. Therefore, a ciphertext encrypted as

$$C_0 = m \cdot Z^s \qquad C_1 = g^s \qquad C_2 = \big(U_0 \cdot \prod_{\ell=1}^{n} U_\ell^{i_\ell}\big)^s,$$

for a random $s \xleftarrow{R} \mathbb{Z}_p^*$, can be deciphered by computing $m = C_0 \cdot e(C_2, d_2)/e(C_1, d_1)$ (this is easily observed by raising both members of (1) to the power s).

Wildcard IBE schemes [1] (or WIBE for short) are hierarchical IBE systems where certain levels of the hierarchy can be left unspecified by a sender willing to allow decryption by any hierarchy member whose identity fits a certain pattern. These WIBE systems were notably used to construct multi-receiver encryption systems. In the case of Waters' IBE, the unique level of the hierarchy can be left unspecified by replacing the ciphertext component C_2 with a vector $(U_0^s, U_1^s, \ldots U_n^s)$ so that $C_2 = \big(U_0 \cdot \prod_{\ell=1}^{n} U_\ell^{i_\ell}\big)^s$ can be reconstructed at decryption for any identity $id \in \{0,1\}^n$. Placing such a "wildcard" at the unique level of the hierarchy permits decryption by anyone holding a decryption key for some identity. The same underlying idea was used in [2] to devise an identity-based traitor tracing scheme from a 2-level WIBE built on [29].

At high level, our scheme can be seen as using a multi-receiver encryption scheme derived from the single level $\mathcal{WA}$-$\mathcal{WIBE}$ of [1]. Instead of assigning a unique identifier to decryption keys as in [2], we embed it in re-encryption keys.

These re-encryption keys are generated by binding decryption keys of the multi-receiver scheme to delegatees' public keys. Identity-based private keys are associated with serial numbers (seen as identities) and tied up to the public keys of entities to whom messages must be re-encrypted. More precisely, we let each party j generate an additional public key component $Y_j = h^{y_j}$ and a delegation from user i to user j is made effective by the re-encryption key

$$R_{ij} = (id, A_{ij}, B_{ij}) = \big(id, Y_j^{z_i} \cdot (V_{i,0} \cdot \prod_{\ell=1}^{n} V_{i,\ell}^{i_\ell})^r, h^r\big)$$

where $pk_i = (Z_i = e(g,h)^{z_i}, Y_i = h^{y_i}, U_{i,0}, \ldots, U_{i,n})$ is user i's public key and $U_{i,\ell} = \psi(V_{i,\ell})$ for $\ell = 0, \ldots, n$. The re-encryption algorithm can actually be thought of as translating WIBE ciphertexts into regular public key encryptions under the delegatee's public key.

The tracing system is non-black-box. It takes as input a pirate re-encryption key and merely extracts the built-in serial number from it. With a non-black-box tracing algorithm, we do not need collusion-secure codes [14]. The proof of traceability takes advantage of the collusion-resistance of the underlying WIBE and we have logarithmic-size ciphertexts in the number of delegations.

3.2 The Scheme

For simplicity, we assume that all users have at most N delegatees. Public keys and second level ciphertexts consist of $O(n) = O(\log N)$ group elements.

Global-setup(λ)**:** on input of a security parameter λ, choose bilinear map groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi)$ of prime order $p > 2^\lambda$ with generators $h \xleftarrow{R} \mathbb{G}_2$, $g = \psi(h)$.

Keygen(λ)**:** user i sets his public key as

$$pk_i = \big(Z_i = e(g,h)^{z_i}, Y_i = h^{y_i}, U_{i,0} = g^{u_{i,0}}, U_{i,1} = g^{u_{i,1}}, \ldots, U_{i,n} = g^{u_{i,n}}\big)$$

for random values $(z_i, y_i, u_{i,0}, u_{i,1}, \ldots, u_{i,n}) \xleftarrow{R} (\mathbb{Z}_p^*)^{n+3}$. For $\ell = 0, \ldots, n$, group elements $V_{i,\ell} = h^{u_{i,\ell}} \in \mathbb{G}_2$ (that satisfy $U_{i,\ell} = \psi(V_{i,\ell})$) are also computed and included in the private key $sk_i = (z_i, y_i, V_{i,0}, \ldots, V_{i,n})$. Let $w_{i,j} \in \{0,1\}^n$ be a unique identifier to be assigned by user i to the re-encryption key R_{ij} translating to user j. Elements $U_{i,\ell}, V_{i,\ell}$ define functions

$$F_{V_i} : \{0,1\}^n \to \mathbb{G}_2 : F_{V_i}(w_{i,j}) = V_{i,0} \cdot \prod_{\ell=1}^{n} V_{i,\ell}^{w_{i,j,\ell}}$$

and $F_{U_i} : \{0,1\}^n \to \mathbb{G}_1 : F_{U_i}(w_{i,j}) = \psi\big(F_{V_i}(w_{i,j})\big)$.

ReKeygen(sk_i, pk_j)**:** given user i's private key $sk_i = (z_i, y_i, V_{i,0}, \ldots, V_{i,n})$ and user j's public key $pk_j = (Z_j, Y_j, U_{j,0}, U_{j,1}, \ldots, U_{j,n})$, choose[4] a previously unemployed string $w_{i,j} = w_{i,j,1} \ldots w_{i,j,n} \in \{0,1\}^n$ and a random exponent $r \xleftarrow{R} \mathbb{Z}_p^*$ to generate the unidirectional key

$$R_{ij} = (w_{i,j}, A_{ij}, B_{ij}) = (w_{i,j}, Y_j^{z_i} \cdot F_{V_i}(w_{i,j})^r, h^r).$$

CheckKey(sk_i, pk_j, R_{ij})**:** given $sk_i = (z_i, y_i, V_{i,0}, \ldots, V_{i,n})$, parse user j's public key pk_j as $(Z_j, Y_j, U_{j,0}, U_{j,1}, \ldots, U_{j,n})$ and R_{ij} as $(w_{i,j}, A_{ij}, B_{ij})$. Return 1 if

$$e(g, A_{ij}) = e(g, Y_j)^{z_i} \cdot e(F_{U_i}(w_{i,j}), B_{ij}) \tag{2}$$

and 0 otherwise.

[4] in order to avoid to store w_{ij} and r, the delegator can compute them as a pseudo-random function of a short secret key and the public key pk_j.

$\mathsf{Enc}_1(m, pk_i, \mathsf{par})$: to encrypt a message $m \in \mathbb{G}_T$ under the public key $pk_i = (Z_i, Y_i, U_{i,0}, U_{i,1}, \ldots, U_{i,n})$ at the first level, choose $s \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and output

$$\mathbf{C} = (C_0, C_1) = \big(m \cdot e(g,h)^s, e(g, Y_i)^s\big)$$

$\mathsf{Enc}_2(m, pk_i, \mathsf{par})$: to encrypt a message $m \in \mathbb{G}_T$ under the public key pk_i at level 2, the sender picks a random exponent $s \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$$\mathbf{C} = (C_0, C_1, C_{2,0}, C_{2,1}, \ldots, C_{2,n}) = \big(m \cdot Z_i^s, g^s, U_{i,0}^s, U_{i,1}^s, \ldots, U_{i,n}^s\big)$$

$\mathsf{ReEnc}(R_{ij}, \mathbf{C}_i)$: given the translation key $R_{ij} = (w_{i,j}, A_{ij}, B_{ij}) \in \{0,1\}^n \times \mathbb{G}_2^2$ and a ciphertext $\mathbf{C}_i = (C_0, C_1, C_{2,0}, \ldots, C_{2,n}) \in \mathbb{G}_T \times \mathbb{G}_1^{n+2}$, compute

$$F_{U_i}(w_{i,j})^s = C_{2,0} \cdot \prod_{\ell=1}^{n} C_{2,\ell}^{w_{i,j,\ell}} = \big(U_{i,0} \cdot \prod_{\ell=1}^{n} U_{i,\ell}^{w_{i,j,\ell}}\big)^s$$

and output

$$\mathbf{C}'_j = (C'_0, C'_1) = \Big(C_0, \frac{e(C_1, A_{ij})}{e(F_{U_i}(w_{i,j})^s, B_{ij})}\Big) \tag{3}$$

$$= \Big(m \cdot e(g,h)^{z_i s}, e(g, Y_j)^{z_i s}\Big) = \Big(m \cdot e(g,h)^{\tilde{s}}, e(g, Y_j)^{\tilde{s}}\Big) \tag{4}$$

with $\tilde{s} = s z_i$.

$\mathsf{Dec}_1(\mathbf{C}_j, sk_j)$: given $sk_j = (z_j, y_j, V_{j,0}, \ldots, V_{j,n})$, parse the ciphertext $\mathbf{C}_j$ as $(C_0, C_1) \in \mathbb{G}_T^2$. Return $m = C_0 / C_1^{1/y_j}$.

$\mathsf{Dec}_2(\mathbf{C}_i, sk_i)$: parse $\mathbf{C}_i$ as $C = (C_0, C_1, C_{2,0}, \ldots, C_{2,n}) \in \mathbb{G}_T \times \mathbb{G}_1^{n+2}$ and sk_i as $(z_i, y_i, V_{i,0}, \ldots, V_{i,n})$. Return $m = C_0 / e(C_1, h)^{z_i}$.

$\mathsf{Trace}(sk_i, R_{it}, pk_t)$: on input of a public key $pk_t = (Z_t, Y_t, U_{t,0}, \ldots, U_{t,n})$ and a re-encryption key $R_{it} = (w, A_{it}, B_{it}) \in \{0,1\}^n \times \mathbb{G}_2 \times \mathbb{G}_2$ such that $\mathsf{CheckKey}(sk_i, pk_t, R_{it}) = 1$, this algorithm incriminates the proxy that has been provided with a re-encryption key including w as identifier.

The correctness of the re-encryption algorithm is easily checked by observing that re-encryption keys $R_{ij} = (w_{i,j}, A_{ij}, B_{ij})$ always satisfy relation (2). Raising both members of the latter to the power $s \in \mathbb{Z}_p^*$ gives

$$e(g^s, A_{ij}) = e(g, Y_j)^{z_i s} \cdot e(F_{U_i}(w_{i,j})^s, B_{ij})$$

which explains the transition between relations (3) and (4).

As in prior unidirectional schemes, the proxy and the delegator can collude to compute and disclose a quantity that allows opening all second level ciphertexts: given $R_{ij} = (w_{i,j}, A_{ij}, B_{ij})$ and y_j s.t. $Y_j = h^{y_j}$, they can obtain

$$R_{i\star}^{\mathsf{bad}} = (w_{i,j}, A_{ij}^{1/y_j}, B_{ij}^{1/y_j}) = \big(w_{i,j}, h^{z_i} \cdot F_{V_i}(w_{i,j})^{r'}, h^{r'}\big),$$

with $r' = r/y_j$, that allows for the off-line decryption of level 2 ciphertexts. However, when presented with $R_{i\star}^{\mathsf{bad}} = (w_{i,j}, A'_{ij}, B'_{ij})$, the tracing algorithm runs the

validity check $e(g, A'_{ij}) \stackrel{?}{=} Z_i \cdot e(F_{U_i}(w_{i,j}), B'_{ij})$. If the latter test is successful, the the proxy identified by $w_{i,j}$ and its associated delegatee are *both* found guilty for having conspired to produce $R_{i\star}^{\mathsf{bad}}$. The serial number $w_{i,j}$ makes the source of the collusion evident and provides a deterrent for abuses of trust.

When the tracing system takes as input a pair $(R_{it} = (w, A_{it}, B_{it}), pk_t)$, the original delegatee j associated the serial number $w = w_{ij}$ cannot be incriminated as the corrupt proxy may have maliciously chosen pk_t as a function of pk_j (possibly in an attempt to trick user i into believing that j is not trustworthy).

3.3 Security

Theorem 1. *The scheme is semantically secure at the second level under the Augmented DBDH assumption.*

Proof. Let $(A = h^a, B = h^b, C = h^c, D = h^{a^2 b}, T) \in \mathbb{G}_2^4 \times \mathbb{G}_T$ be an Augmented DBDH instance. We construct an algorithm $\mathcal{B}$ that decides if $T = e(g, h)^{abc}$ using its interaction with a chosen-plaintext adversary $\mathcal{A}$.

All public keys that $\mathcal{A}$ gets to see are indexed by an integer $i \in \{0, \ldots, N_{max}\}$, where $N_{max} + 1$ denotes the maximal number of users in the system. Let us call $HU \subset \{0, \ldots, N_{max}\}$ the set of honest players, including the target receiver whose public key has index 0. Let also $CU \subset \{1, \ldots, N_{max}\}$ denote the set of corrupt receivers. The attack environment is emulated as follows.

- *Key generation*:
  - The public key $pk_0 = (Z_0, Y_0, U_{0,0}, U_{0,1}, \ldots, U_{0,n})$ of the target user is chosen as $Z_0 = e(\psi(A), B) = e(g, h)^{ab}$ and $Y_0 = h^{y_0}$, $U_{0,\ell} = g^{u_{0,\ell}}$ with $y_0, u_{0,\ell} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ for $\ell = 0, \ldots, n$.
  - For users $i \in HU \backslash \{0\}$, public keys are defined by randomly choosing $z_i, y_i, u_{i,0}, \ldots, u_{i,n} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and setting $Z_i = e(g, h)^{z_i}$, $Y_i = A^{y_i} = h^{a y_i}$ and $U_{i,\ell} = g^{u_{i,\ell}}$ for $\ell = 0, \ldots, n$.
  - For corrupt users $i \in CU$, $\mathcal{B}$ generates pk_i according to the specification of the scheme and discloses private elements $z_i, y_i, u_{i,0}, \ldots, u_{i,n} \in \mathbb{Z}_p^*$.
- *Re-encryption key generation*: to generate re-encryption keys R_{ij} from player i to player j, $\mathcal{B}$ has to distinguish several situations.
  - If $i \in CU$ or $i \in HU \backslash \{0\}$, $\mathcal{B}$ knows user i's private key component z_i such that $Z_i = e(g, h)^{z_i}$ and generates a re-encryption key as specified by the re-encryption algorithm.
  - If $i = 0$ and $j \in HU \backslash \{0\}$, $\mathcal{B}$ picks a new string $w_{0,j} \in \{0,1\}^n$ and a random exponent $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ to return

$$R_{0j} = \Big(w_{0,j}, D^{y_j} \cdot F_{V_0}(w_{0,j})^r, h^r\Big),$$

    for a random $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Observe that R_{0j} has the correct shape since $Z_0 = e(g, h)^{ab}$, $Y_j = A^{y_j} = h^{a y_j}$ and $D^{y_j} = (h^{a^2 b y_j}) = (h^{a y_j})^{ab}$.

- *Challenge*: when $\mathcal{A}$ comes up with messages $m_0, m_1 \in \mathbb{G}_T$, $\mathcal{B}$ flips a fair coin $d^\star \xleftarrow{R} \{0,1\}$ and sets the challenge ciphertext as

$$C_0 = m_{d^\star} \cdot T \qquad C_1 = \psi(C) \qquad C_{2,\ell} = \psi(C)^{u_{0,\ell}} \quad \text{for } \ell = 0, \ldots, n.$$

  Since $C = h^c$ and $Z_0 = e(g,h)^{ab}$, $\mathbf{C} = (C_0, C_1, C_{2,0}, \ldots, C_{2,n})$ is a valid encryption of $m_{d^\star}$ under pk_0 with the encryption exponent $s = c$ whenever $T = e(g,h)^{abc}$. When T is random in $\mathbb{G}_T$, $\mathbf{C}$ leaks no information on $d^\star$ and $\mathcal{A}$ can only guess it with probability $1/2$. Therefore, $\mathcal{B}$ outputs 1 (meaning that $T = e(g,h)^{abc}$) if $\mathcal{A}$ successfully guesses $d^\star$ and 0 otherwise. □

Theorem 2. *The scheme is semantically secure at the first level under the DBDH assumption.*

Proof. Given in appendix A. □

Theorem 3. *The scheme is traceable under the* 2-3-*CDH assumption in* $\mathbb{G}_2$.

Proof. For the sake of contradiction, assume that an adversary $\mathcal{A}$ defeats the non-black-box tracing algorithm (in the sense of definition 3) with probability ε. We build an algorithm $\mathcal{B}''$ solving a 2-3-CDH instance $(A = h^a, B = h^b)$ with probability $O(\varepsilon/q_{rk})$, where q_{rk} is the number of re-encryption key queries.

- *Key generation*: a set of public keys is prepared by $\mathcal{B}''$. For the target user 0, it first defines $Z_0 = e(\psi(A), B) = e(g,h)^{ab}$ and $Y_0 = h^{y_0}$ for a random $y_0 \xleftarrow{R} \mathbb{Z}_p^*$. The vector $(V_{0,0}, V_{0,1}, \ldots, V_{0,n})$ is defined as $V_{0,0} = A^{\alpha_0 - \kappa\tau} \cdot h^{\beta_0}$, $V_{0,\ell} = A^{\alpha_\ell} \cdot h^{\beta_\ell}$ for $\ell \in \{1, \ldots, n\}$ using random vectors $(\alpha_0, \alpha_1, \ldots, \alpha_n) \xleftarrow{R} \mathbb{Z}_\tau^{n+1}$, $(\beta_0, \beta_1, \ldots, \beta_n) \xleftarrow{R} \mathbb{Z}_p^{n+1}$, where $\kappa \xleftarrow{R} \{0, \ldots, n\}$ is chosen at random and $\tau = 2q_{rk}$. For any string $w_{0,j} = w_{0,j,1} \ldots w_{0,j,n} \in \{0,1\}^n$, we have

$$F_{V_0}(w_{0,j}) = V' \cdot \prod_{\ell=1}^{n} V_{0,\ell}^{w_{0,j,\ell}} = A^{J(w_{0,j})} h^{K(w_{0,j})}$$

  for functions $J : \{0,1\}^n \to \mathbb{Z}$, $K : \{0,1\}^n \to \mathbb{Z}_p$ respectively defined as $J(w_{0,j}) = \alpha_0 + \sum_{\ell=1}^{n} \alpha_\ell w_{0,j,\ell} - \kappa\tau$ and $K(w_{0,j}) = \beta_0 + \sum_{\ell=1}^{n} \beta_\ell w_{0,j,\ell}$. For $\ell = 0, \ldots, n$, $\mathcal{B}''$ also sets $U_{0,\ell} = \psi(V_{0,\ell})$. As in [29], the simulator will be successful if $J(w_{0,j}) \neq 0$ for all strings $w_{0,j} \neq w^\star$ involved in delegation queries whereas $J(w^\star) = 0$ for the identifier $w^\star$ of the re-encryption key produced by $\mathcal{A}$ at the tracing stage. Since $|J(.)| \leq \tau(n+1) \ll p$, we have $J(w^\star) = 0$ with non-negligible probability $O(1/\tau(n+1))$. For all other (honest or corrupt) users $i \in \{1, \ldots, N_{max}\}$, public keys are honestly generated by $\mathcal{B}''$ that chooses the private keys $(z_i, y_i, u_{i,0}, \ldots, u_{i,n}) \in \mathbb{Z}_p^{n+3}$. The latter secrets are given to $\mathcal{A}$ for indices $i \in CU \subset \{1, \ldots, N_{max}\}$ of corrupt users.
- *Re-encryption key queries*: at any time, $\mathcal{A}$ may ask for re-encryption keys R_{ij} of her choosing. When $i \neq 0$, $\mathcal{B}''$ knows user i's private key and can normally handle the delegation query. Otherwise, following the technique of

[9, 29], it constructs a re-encryption key by sampling a fresh random string $w_{0,j} \stackrel{R}{\leftarrow} \{0,1\}^n$ and a random exponent $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$ to compute

$$R_{0j} = (w_{0,j}, A_{0j}, B_{0j}) = \left(w_{0,j}, B^{-y_j \frac{K(w_{0,j})}{J(w_{0,j})}} \cdot F_{V_0}(w_{0,j})^r, B^{-\frac{y_j}{J(w_{0,j})}} \cdot h^r \right),$$

where $y_j \in \mathbb{Z}_p^*$ is part of user j's private key, which is returned to $\mathcal{A}$. If we define $\tilde{r} = r - (by_j)/J(w_{0,j})$, R_{0j} has the correct distribution since

$$\begin{aligned} A_{0j} &= B^{-y_j \frac{K(w_{0,j})}{J(w_{0,j})}} \cdot F(w_{0,j})^r \\ &= B^{-y_j \frac{K(w_{0,j})}{J(w_{0,j})}} \cdot F(w_{0,j})^{\tilde{r}} \cdot (A^{J(w_{0,j})} \cdot h^{K(w_{0,j})})^{\frac{by_j}{J(w_{0,j})}} = (h^{y_j})^{ab} \cdot F(w_{0,j})^{\tilde{r}} \end{aligned}$$

and $B_{0j} = h^{\tilde{r}}$. If $J(w_{0,j}) = 0$, $\mathcal{B}''$ aborts as it cannot answer the query.

- *Tracing stage*: a successful attacker must output a pair $(R_{0t}^\star, pk_t)$ such that $\mathsf{CheckKey}(sk_0, R_{0t}^\star, pk_t) = 1$ and $R_{0t}^\star = (w^\star, A_{0t}^\star, B_{0t}^\star)$ cannot be traced to a member of the coalition T. This implies that $w^\star$ must differ from all the serial numbers w_{0j} that were associated with user 0's delegatees. At this point, $\mathcal{B}''$ declares failure if $J(w^\star) \neq 0$. With probability at least $1/4q_{rk}(n+1)$ (see [29] for a detailed analysis of this probability) such a failure state is avoided. In this case, $\mathcal{B}''$ parses pk_t as $(Z_t, Y_t, U_{t,0}, \ldots, U_{t,n})$ and outputs

$$\left(Y_t, A_{0t}^\star / {B_{0t}^\star}^{K(w^\star)}\right) = (Y_t, Y_t^{ab})$$

which solves the 2-3-CDH problem in $\mathbb{G}_2$. □

4 A Variant with Black Box k-Traceability

The scheme can be endowed with a black-box tracing mechanism which is similar to the one described in [2]. The idea is to associate identity-based private keys with the codewords (seen as identities) of a collusion-secure code [14] instead of serial numbers. These keys are bound to delegatees' public keys to form fingerprinted re-encryption keys. Assuming the hardness of the Decision Diffie-Hellman problem in $\mathbb{G}_1$ for configurations where $\mathbb{G}_1 \neq \mathbb{G}_2$ (and no isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ is computable), well-formed ciphertexts are not publicly recognizable. Then, pirate re-encryption devices $\mathbb{P}$ can be probed with invalid ciphertexts so as to determine the codeword of one of the pirate re-encryption keys.

As in [2], this comes at the expense of prohibitively large ciphertexts, the size of which becomes proportional to the length of the collusion-secure code. We need a binary (k, N, ε)-collusion-secure code (as defined in appendix B), where N is the the maximal number of delegatees per user, k is the maximal number of colluding proxies against a delegator and ε is the maximal probability that a colluder avoids being traced. Such a code can be obtained with codewords of length $n = O\big(k^2(\log N + \log(\varepsilon^{-1}))\big)$ [28], which is also the number of group elements in a ciphertext. If users have at most $N = 100$ delegatees, in the case $k \approx 10$, we end up with ciphertexts made of about 700 group elements (which

amounts to 13 Kb using curves [6] where elements of $\mathbb{G}_1$ have a 161-bit representation). We leave open the problem of constructing an efficient black-box traceable scheme.

The tracing system, borrowed from [2], probes re-encryption devices with second level ciphertexts wherein certain components have been altered and eventually retrieves bits at all positions where words in the feasible set of the coalition (see appendices B and C for details) are identical. More precisely, the tracing algorithm checks whether the pirate device successfully re-encrypts ciphertexts where components $C_{2,\ell}$ (for all $\ell \in \{1, \ldots, n\}$), have been tampered with. If it does, the tracer deduces that $C_{2,\ell}$ was not used by the pirate device, which means that the associated bit is 0 in all codewords that were assigned to re-encryption keys available to the coalition. Once a n-bit word in the feasible set of the coalition has been found, the tracing procedure of the collusion-secure code allows recovering the fingerprint of one of the involved re-encryption keys, which identifies a misbehaving proxy.

5 Conclusion

In all PRE schemes proposed so far, proxies and delegatees can derive new re-encryption keys for receivers that are not trusted by the delegator. In this paper, we proposed traceable proxy re-encryption systems, in which proxies that leak their re-encryption key can be identified by the delegator and we presented an efficient realization of this concept. An interesting open issue is to design a more efficient TPRE scheme with black-box traceability.

Acknowledgements. We thank Duong Hieu Phan for his comments. The first author is supported by the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.). The second author is supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by the French *Agence Nationale de la Recherche* through the PACE project.

References

1. M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, N. Smart. Identity-Based Encryption Gone Wild. In *ICALP'06*, *LNCS* 4052, pp. 300–311. Springer, 2006.
2. M. Abdalla, A. Dent, J. Malone-Lee, G. Neven, D. Phan, N. Smart. Identity-Based Traitor Tracing. In *PKC'07*, *LNCS* 4450, pp. 361–376. Springer, 2007.
3. S. Al-Riyami, K. Paterson. Certificateless Public Key Cryptography. In *Asiacrypt'03*, *LNCS* 2894, pp. 452–473. Springer, 2003.
4. G. Ateniese, K. Fu, M. Green, S. Hohenberger. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In *NDSS*, 2005.
5. —. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ACM TISSEC*, 9(1): pp. 1–30, 2006.
6. P. S. L. M. Barreto, M. Naehrig. Pairing-Friendly Elliptic Curves of Prime Order. In *SAC'05. LNCS* 3897, pp. 319–331. Springer, 2006.

7. M. Bellare, P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security*, pp. 62–73, ACM Press, 1993.
8. M. Blaze, G. Bleumer, M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *Eurocrypt'98*, *LNCS* 1403, pp. 127–144, 1998.
9. D. Boneh, X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Eurocrypt'04*, *LNCS* 3027, pp. 223–238. Springer, 2004.
10. D. Boneh, X. Boyen, E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Eurocrypt'05*, *LNCS* 3494, pp. 440–456. Springer, 2005.
11. D. Boneh, M. Franklin. Identity-based encryption from the Weil pairing. In *Crypto'01*, *LNCS* 2139, pp. 213–229. Springer, 2001.
12. D. Boneh, C. Gentry, B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Crypto'05*, *LNCS* 3621, pp. 258–275. Springer, 2005
13. D. Boneh, A. Sahai, B. Waters. Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In *Eurocrypt'06*, *LNCS* 4004, pp. 573–592. Springer, 2006.
14. D. Boneh, J. Shaw. Collusion-Secure Fingerprinting for Digital Data. In *Crypto'95*, *LNCS* 936, pp. 452–465. Springer, 1995.
15. J. Camenisch, S. Hohenberger and A. Lysyanskaya. Compact E-Cash. In *Eurocrypt'05*, *LNCS* 3494, pp. 302–321, Springer, 2005.
16. R. Canetti, S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In *ACM CCS'07*, pp. 185–194. ACM Press, 2007.
17. B. Chor, A. Fiat, M. Naor. Tracing Traitors. In *Crypto'94*, *LNCS* 839, pp. 257–270. Springer, 1994.
18. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Crypto'84*, *LNCS* 196, pp. 10–18. Springer, 1985.
19. M. Green, G. Ateniese. Identity-Based Proxy Re-encryption. In *ACNS'07*, *LNCS* 4521, pp. 288–306. Springer, 2007.
20. V. Goyal. Reducing Trust in the PKG in Identity Based Cryptosystems. In *Crypto'07*, *LNCS* 4622, pp. 430–447. Springer, 2007.
21. S. Hohenberger. Advances in Signatures, Encryption, and E-Cash from Bilinear Groups. Ph.D. Thesis, MIT, May 2006.
22. S. Hohenberger, G. N. Rothblum, a. shelat, V. Vaikuntanathan. Securely Obfuscating Re-encryption. In *TCC'07*, *LNCS* 4392, pp. 233–252. Springer, 2007.
23. S. Kunz-Jacques, D. Pointcheval. About the Security of MTI/C0 and MQV. In *SCN'06*, *LNCS* 4116, pp. 156–172, Springer, 2006.
24. B. Libert, D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption. In *PKC'08*, *LNCS* 4939, pp. 360–379, Springer, 2008.
25. T. Matsuo. Proxy Re-encryption Systems for Identity-based Encryption. In *Pairing'07*, *LNCS* 4575, pp. 247–267. Springer, 2007.
26. M. Scott. Authenticated ID-based Key Exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive: Report 2002/164, 2002.
27. A. Shamir. Identity based cryptosystems and signature schemes. In *Crypto'84*, *LNCS* 196, pp. 47–53. Springer, 1984.
28. G. Tardos. Optimal probabilistic fingerprint codes. In *STOC'03*, pp. 116–125. ACM Press, 2003.
29. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Eurocrypt'05*, *LNCS* 3494, pp. 114–127. Springer 2005.

A Proof of Theorem 2

Let $(A = h^a, B = h^b, C = h^c, T \stackrel{?}{=} e(g,h)^{abc})$ be a DBDH instance. We show a simple distinguisher $\mathcal{B}'$ built from an adversary $\mathcal{A}$ against first level challenge ciphertexts. For the target user, the public key pk_0 is made of $Z_0 = e(g,h)^{z_0}$, $Y_0 = C = h^c$, $U_{0,\ell} = g^{u_{0,\ell}}$ for $\ell = 0, \ldots, n$ with $z_0, u_{0,0}, \ldots, u_{0,n} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. All other users' public keys are honestly generated and $\mathcal{B}'$ knows the corresponding secret key $sk_i = (z_i, y_i, u_{i,0}, \ldots, u_{i,n})$. Recall that all re-encryption keys must be given to the adversary. Since $\mathcal{B}'$ knows $z_i \in \mathbb{Z}_p$ such that $Z_i = e(g,h)^{z_i}$ for all users (including user 0), it can handle all delegation queries on behalf of all parties acting as delegators.

At the challenge step, $\mathcal{A}$ outputs messages $m_0, m_1 \in \mathbb{G}_T$ and expects to receive a challenge ciphertext encrypted for user 0. To generate it, $\mathcal{B}'$ flips a fair coin $d^\star \stackrel{R}{\leftarrow} \{0,1\}$ and sets

$$C_0 = m_{d^\star} \cdot e(\psi(A), B) \qquad C_1 = T.$$

Since $Y_0 = h^c$, it can be readily observed that $\mathbf{C} = (C_0, C_1)$ is a proper encryption of $m_{d^\star}$ with the encryption exponent $s = ab$ if $T = e(g,h)^{abc}$. If T is random, the bit $d^\star$ is perfectly hidden from $\mathcal{A}$. As usual, $\mathcal{B}'$ decides that $T = e(g,h)^{abc}$ if and only if $\mathcal{A}$'s guess is correct. □

B Binary Collusion-Secure (Fingerprinting) Codes

In order to make the description of the scheme with black-box traceability self-contained, we review in this appendix the definition of *collusion-secure (fingerprinting) codes* from [14]. We only consider *binary codes* (*i.e.* codes defined over $\{0,1\}$) and for more details on collusion-secure codes, we refer the reader to [14, 28] and references therein.

We begin by defining some notation:

- $x \in \{0,1\}^n$ is called a *binary word of length* n. For such a word, we write $x = x_1 \ldots x_n$ where $x_i \in \{0,1\}$ is the i^{th} bit of x (for $i \in \{1, \ldots, n\}$).
- Let $I = \{1 \leq i_1 < \ldots < i_j \leq n\}$ be a set of indices. For a word $x \in \{0,1\}^n$, $x_{|I}$ denotes the subword $x_{i_1} \ldots x_{i_j} \in \{0,1\}^n$ made of bits at positions in I.
- Let $W = \{w_1, \ldots, w_j \in \{0,1\}^n\}$ be a set of words, 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 $w_{1|I} = \cdots = w_{k|I}$. The *feasible set* $FS(W)$ of W is defined as the set of all strings that are equal to $w_1, \ldots, w_k$ at positions in I, *i.e.*

$$FS(W) = \{x \in \{0,1\}^n : x_{|I} = w_{1|I} = \cdots = w_{k|I}\}.$$

The formal definition of collusion-secure codes proposed by Boneh and Shaw in [14] is the following:

Definition 6. *Let $0 < k \leq N$ be positive integers and $\varepsilon \in (0, 1]$. A* binary (k, N, ε) collusion-secure code of length n *consists of a tracing algorithm $\mathcal{T}$, a set $\mathbb{C}$ called the* codebook*, of indexed* codeswords w_i *for $1 \leq i \leq N$ and a trapdoor τ. These are such that for all collusions $C \subset \{1, \ldots, N\}$ of size at most k, $W = \{w_i : i \in C\}$, and for all (unbounded) algorithms $\mathcal{A}$ it holds that*

$$\Pr\left[\mathcal{T}(x, \tau) \in C | x \in FS(W); x \leftarrow \mathcal{A}(W)\right] > 1 - \varepsilon,$$

where the probability is taken over the random coins of $\mathcal{T}$ and $\mathcal{A}$.

C Details of the Scheme with Black-Box Tracing

The variant with black-box traceability is very close to the scheme of section 3 and we just outline the simple modifications that are required.

As in [2], we assume that pirate devices do not retain state information from prior re-encryptions when run in tracing mode.

Unlike what occurs in the scheme of section 3, the black-box tracing algorithm does not allow to incriminate delegatees when we run it on input of a pirate subkey that decrypts at level 2. The reason is that the reconstructed word eventually lies in the feasible set of codewords assigned to *all* re-encryption keys (i.e. those assigned to dishonest delegatees as well as those corresponding to honest ones) that were made available to the coalition.

Global-setup(λ)**:** is the same as in section 3.2.

Keygen(λ)**:** is as in section 3.2 with the difference that user i also selects a set $\mathbb{C}_i$ of N binary words $w_{i,1}, \ldots, w_{i,N}$ of length n that form a (k, N, ε) collusion-secure code. The latter is generated with an underlying trapdoor τ_i to be used by its tracing procedure and that is also part of user i's private key. For codewords, elements $U_{i,\ell}, V_{i,\ell}$ define functions $F_{V_i} : \{0,1\}^n \to \mathbb{G}_2$ and $F_{U_i} : \{0,1\}^n \to \mathbb{G}_1$ as in section 3.2.

ReKeygen, **Enc**$_2$, **Enc**$_1$, **ReEnc**, **Dec**$_2$ and **Dec**$_1$ also remain unchanged.

Trace$(sk_i, \mathbb{P})$**:** given oracle access to a pirate proxy $\mathbb{P}$ that correctly re-encrypts with probability δ, the tracing algorithm conducts the following steps.

Let $pk_t = (Z_t, Y_t, U_{t,0}, \ldots, U_{t,n})$ be the public key under which $\mathbb{P}$ re-encrypts ciphertexts. For $\ell = 1, \ldots, n$, initialize a counter $ctr_\ell \leftarrow 0$ and run the following test $L = 16\lambda/\delta$ times:

1. Choose a random message $m \in \mathbb{G}_T$ and encrypt it using a random exponent $s \xleftarrow{R} \mathbb{Z}_p^*$ to get a ciphertext $\mathbf{C} = (C_0, C_1, C_{2,0}, C_{2,1}, \ldots, C_{2,n})$.
2. Replace element $C_{2,\ell}$ with a random element from $\mathbb{G}_1$.
3. Query the pirate proxy $\mathbb{P}$ on the altered ciphertext.
4. If $\mathbb{P}$ actually re-encrypts $\mathbf{C}$ as a first level ciphertext $\mathbf{C}' = (C_0, C_1')$ with $C_1' = e(g^s, Y_t)^{z_i}$, increase ctr_ℓ.

After these L iterations, set $w_\ell^{\mathbb{P}} \leftarrow 1$ if $ctr_\ell < 4\lambda$ and $w_\ell^{\mathbb{P}} \leftarrow 0$ otherwise.

The decoded n-bit word $w^{\mathbb{P}}$ is finally taken as input by the tracing procedure of the collusion-secure code that uses the trapdoor τ_i to uncover the identity of a rogue proxy with probability ε.

If I denotes the set of positions where all codewords of the coalition are identical, bits of $w^{\mathbb{P}}$ outside I can be arbitrarily chosen by the pirate device (that can notice the ill-formedness of the ciphertext when its altered component is $C_{2,\ell}$ for $\ell \notin I$). But it does not matter since, as in [2], the tracing system of the code only needs a word $w^{\mathbb{P}} \in \{0,1\}^n$ *inside* the feasible set.

It is essentially routine to prove the black-box traceability property using ideas from [2] but a slightly different assumption is needed. As in [2], we first have to count on the difficulty of DDH in $\mathbb{G}_1$ within asymmetric pairing configuration. This assumption obviously requires the infeasibility of inverting $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ and found several applications (see [26, 15, 2] for instance).

Definition 7. *The **eXternal Diffie-Hellman assumption** (XDH) in asymmetric bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ posits the hardness of the Decisional Diffie-Hellman problem in $\mathbb{G}_1$: given $(g^a, g^b) \in {\mathbb{G}_1}^2$, distinguishing g^{ab} from random should be hard. A distinguisher's advantage can be defined as in definition 4.*

The second assumption that we make is a generalization – introduced in [3] – of the computational BDH assumption (CBDH).

Definition 8. *The **Generalized Bilinear Diffie-Hellman Problem** (GBDH) is, given $(h^a, h^b, h^c) \in \mathbb{G}_2^3$, to come up with a pair $(g', e(g', h)^{abc}) \in \mathbb{G}_1 \times \mathbb{G}_T$.*

The GBDH assumption is non-standard but it is worth mentioning that any algorithm breaking it would also be able to solve the Decision Tripartite Diffie-Hellman problem in $\mathbb{G}_2$ which is to distinguish h^{abc} from random given (h^a, h^b, h^c) and that has been more widely used (see [13] for instance).

Theorem 4. *The modified scheme is black-box k-traceable assuming that the code is a (k, N, ε)-collusion-secure code of length n, that the XDH assumption holds in $\mathbb{G}_1$ and that the GBDH problem is hard. More concretely, the advantage of any PPT adversary $\mathcal{A}$ in constructing an untraceable re-encryption device that translates ciphertexts with probability δ after having obtained k re-encryption keys is at most*

$$Adv(\mathcal{A})^{\mathsf{TPRE}} \leq \varepsilon + n \cdot (Adv^{\mathsf{GDBH}}(\mathcal{B}'') + \exp(-\lambda))$$

if $\delta > 2 \cdot Adv^{(}\mathcal{B}')^{\mathsf{XDH}}$ where $\mathcal{B}'$, $\mathcal{B}''$ are PPT algorithm that are built on $\mathcal{A}$.

Proof. Given an adversary $\mathcal{A}$ that outputs a pirate device $\mathbb{P}$ translating ciphertexts with probability δ, we construct an attacker $\mathcal{A}'$ against the collusion-secure code. The latter adversary takes a set of k codewords and outputs a new one w'. As in [2], we show that, with all but negligible probability, $\mathcal{A}'$ avoids being traced whenever $\mathcal{A}$ does. Algorithm $\mathcal{A}'$ takes as input a set of random codewords $W = \{w_1, \ldots, w_k\}$ and generates public keys on behalf of all honest and corrupt

users $i \in HU \cup CU$. Codewords of W are used to define the target user's codebook while $\mathcal{A}'$ generates itself the codebooks that are part of other users' private keys. At the j^{th} re-encryption key of the shape (pk_0, pk_j) (i.e. involving user 0 as a delegator and pk_j as a delegatee's public key), $\mathcal{A}'$ fetches a fresh codeword from W and assigns it to the re-encryption key R_{0j} which is returned to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs a pirate translation device $\mathbb{P}$ which is run in tracing mode so as to finally reconstruct a n-bit word w'. As in [2], it can be shown that w' falls outside $FS(W)$ with probability smaller than

$$n \cdot (Adv^{\mathsf{GDBH}}(\mathcal{B}'') + \exp(-\lambda)). \tag{5}$$

Let I be the set of positions that are identical in all words of W. For indices $\ell^\star \in I$ such that $w_{\ell^\star} = 0$, lemma 1 first shows that $\mathbb{P}$ re-encrypts ciphertexts where $C_{2,\ell^\star}$ is random with probability negligibly close to δ unless the XDH assumption is false. For indices $\ell^\star \in I$ where $w_{\ell^\star} = 1$, lemma 2 gives an upper bound on $\mathbb{P}$'s chance to succeed in translating ciphertexts where $C_{2,\ell^\star}$ is perturbed. The claimed bound (5) is obtained through a similar analysis to [2]. □

Lemma 1. *For any $\ell^\star \in \{1, \ldots, n\}$, if $w_{0,j,\ell^\star} = 0$ in all codewords $w_{0,j}$ associated with re-encryption keys available to the coalition, $\mathbb{P}$ has probability at least $p_0 \geq \delta - Adv^{XDH}(\lambda)$ to re-encrypt ciphertexts where $C_{2,\ell^\star}$ was tampered with.*

Proof. Towards a contradiction, assume that an adversary $\mathcal{A}$ comes up with a pirate device $\mathbb{P}$, where $w_{0,j,\ell^\star} = 0$ in all underlying codewords $w_{0,j}$, that re-encrypts ciphertexts with probability $p_0 \leq \delta - \gamma$ for some $\gamma > 0$. Then, there exists an algorithm $\mathcal{B}'$ breaking the XDH assumption with advantage γ.

On input of an XDH instance $(A = g^a, B = g^b, \eta \stackrel{?}{=} g^{ab})$, this algorithm $\mathcal{B}'$ first prepares a set of public keys by defining the target user's public key $pk_0 = (Z_0, Y_0, U_{0,0}, U_{0,1}, \ldots, U_{0,n})$ as $Z_0 = e(g,h)^{z_0}$, $Y_0 = h^{y_0}$, with $z_0, y_0 \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, $U_{0,\ell^\star} = A = g^a$ and $U_{0,\ell} = g^{u_{0,\ell}}$ with $u_{0,\ell} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ for $\ell \in \{0, \ldots, n\} \backslash \{\ell^\star\}$. Note that pre-images $V_{0,\ell} = h^{u_{0,\ell}}$ so that $\psi(V_{0,\ell}) = U_{0,\ell}$ are also available for all $\ell \in \{0, \ldots, n\} \backslash \{\ell^\star\}$. For other public keys pk_i with $i \in \{1, \ldots, N\}$, $\mathcal{B}'$ simply runs the key generation algorithm according to its specification.

As $w_{0,j,\ell^\star} = 0$ for all codewords $w_{0,j}$ assigned to re-encryption keys R_{0j} queried by $\mathcal{A}$, $\mathcal{B}'$ is able to compute such keys $R_{0j} = (w_{0,j}, Y_j^{z_0} \cdot F_{V_0}(w_{0,j})^r, h^r)$ by running $\mathsf{ReKeygen}$ (although it does not know $V_{0,\ell^\star} = \psi^{-1}(g^a)$). When $\mathcal{A}$ outputs a pirate ciphertext translator $\mathbb{P}$, $\mathcal{B}'$ feeds it with a ciphertext

$$C_0 = m \cdot e(B,h)^{z_0} \quad C_1 = B \quad C_{2,\ell^\star} = \eta \quad C_{2,\ell} = B^{u_{0,\ell}} \text{ for } \ell \in \{0, \ldots, n\} \backslash \{\ell^\star\}$$

for a random message $m \stackrel{R}{\leftarrow} \mathbb{G}_T$. The device $\mathbb{P}$ then generates a re-encryption $\mathbf{C}' = (C_0, C_1')$. Given the public key $pk_t = (Z_t, Y_t, U_{t,0}, \ldots, U_{t,n})$ of the user receiving re-encryptions from $\mathbb{P}$, $\mathcal{B}'$ can check whether $\mathbf{C}'$ was successfully translated by testing if $C_1' = e(B, Y_t)^{z_0}$. If yes, $\mathcal{B}'$ outputs 1 (meaning that $\eta = g^{ab}$). Otherwise, it returns 0 and bets that η is random. □

Lemma 2. *For any $\ell^\star \in \{1, \ldots, n\}$, if $w_{0,j,\ell^\star} = 1$ in all codewords $w_{0,j}$ embedded in re-encryption keys of colluding proxies, then $\mathbb{P}$ has probability at most $p_1 \leq Adv^{GBDH}(\lambda)$ to re-encrypt ciphertexts where $C_{2,\ell^\star}$ was tampered with.*

Proof. Assume that $\mathcal{A}$ is an adversary producing a re-encryption box $\mathbb{P}$ that has non-negligible probability p_1 of re-encrypting ciphertexts where $C_{2,\ell^\star}$ has been replaced by a random element of $\mathbb{G}_1$. We construct a distinguisher $\mathcal{B}''$ solving a computational GBDH instance $(A = h^a, B = h^b, C = h^c)$.

$\mathcal{B}''$ first generates a set of public keys. The target user's public key is set as $pk_0 = (Z_0, Y_0, U_{0,0}, U_{0,1}, \ldots, U_{0,n})$ where $Z_0 = e(\psi(A), B) = e(g,h)^{ab}$, $Y_0 = h^{y_0}$ and $U_{0,\ell} = g^{u_{0,\ell}}$ with $y_0, u_{0,\ell} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ for $\ell \in \{0,\ldots,n\}\backslash\{\ell^\star\}$. The remaining public key component is chosen as $U_{0,\ell^\star} = g^{\alpha_{0,\ell^\star}} \cdot \psi(A)^{\beta_{0,\ell^\star}}$ for random integers $\alpha_{0,\ell^\star}, \beta_{0,\ell^\star} \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. Note that $V_{0,\ell^\star} = h^{\alpha_{0,\ell^\star}} \cdot A^{\beta_{0,\ell^\star}}$ is also computable as well as $V_{0,\ell} = h^{u_{0,\ell}}$ for $\ell \neq \ell^\star$. For other users $i \in \{1,\ldots,n\}$, public keys $pk_i = (Z_i, Y_i, U_{i,0}, U_{i,1}, \ldots, U_{i,n})$ are calculated as specified by the key generation algorithm and private elements $(z_i, y_i, u_{i,0}, \ldots, u_{i,n})$ are known to $\mathcal{B}''$.

Given that $w_{0,j,\ell^\star} = 1$ for all of the k codewords $w_{0,j}$ contained in re-encryption keys R_{0j} that $\mathcal{A}$ must be provided with, these keys can be generated by choosing $r \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$ and setting

$$A_{0j} = V_{0,\ell^\star}^r \cdot B^{-\frac{y_j \alpha_{0,\ell^\star}}{\beta_{0,\ell^\star}}} \cdot \prod_{\ell=0,\ell\neq\ell^\star}^{n} \left(V_{0,\ell}^r \cdot B^{-\frac{y_j u_{0,\ell}}{\beta_{0,\ell^\star}}}\right)^{w_{0,j,\ell}}, \qquad B_{0j} = h^r \cdot B^{-\frac{y_j}{\beta_{0,\ell^\star}}}$$

which provides a valid re-encryption key $R_{0j} = (w_{0,j}, A_{0j}, B_{0j})$ since $X_j = h^{x_j}$ and, if we define $\tilde{r} = r - by_j/\beta_{0,\ell^\star}$, we have $B_{0j} = h^{\tilde{r}}$ and

$$\begin{aligned} A_{0j} &= V_{0,\ell^\star}^{\tilde{r}} \cdot (h^{\alpha_{0,\ell^\star}} \cdot A^{\beta_{0,\ell^\star}})^{\frac{by_j}{\beta_{0,\ell^\star}}} \cdot B^{\frac{y_j\alpha_{0,\ell^\star}}{\beta_{0,\ell^\star}}} \cdot \prod_{\ell=0,\ell\neq\ell^\star}^{n} \left(V_{0,\ell}^{\tilde{r}} \cdot h^{u_{0,\ell}\frac{by_j}{\beta_{0,\ell^\star}}} \cdot B^{-\frac{y_j u_{0,\ell}}{\beta_{0,\ell^\star}}}\right)^{w_{0,j,\ell}} \\ &= (h^{y_j})^{ab} \cdot V_{0,\ell^\star}^{\tilde{r}} \cdot \prod_{\ell=0,\ell\neq\ell^\star}^{n} \left(V_{0,\ell}^{\tilde{r}}\right)^{w_{0,j,\ell}}. \end{aligned}$$

When $\mathcal{B}''$ obtains a pirate device $\mathbb{P}$ from $\mathcal{A}$, it probes it with a ciphertext

$$C_0 \stackrel{R}{\leftarrow} \mathbb{G}_T \quad C_1 = \psi(C) \quad C_{2,\ell^\star} \stackrel{R}{\leftarrow} \mathbb{G}_1 \quad C_{2,\ell} = \psi(C)^{u_{0,\ell}} \text{ for } \ell \in \{0,\ldots,n\}\backslash\{\ell^\star\}$$

which is a valid ciphertext (with the encryption exponent $s = c$) where $C_{2,\ell^\star}$ has been replaced by a random element. By assumption, $\mathbb{P}$ is assumed to re-encrypt it under some public key $pk_t = (X_t, Y_t, U_{t,0}, U_{t,1}, \ldots, U_{t,n})$ that was not involved in a re-encryption key query with user 0 acting as a delegator. When obtaining a re-encryption $\mathbf{C}_t' = (C_0, C_1') = \left(C_0, e(g, Y_t)^{abc}\right) = \left(C_0, e(\psi(Y_t), h)^{abc}\right)$, $\mathcal{B}''$ outputs a pair $(\psi(Y_t), C_1')$ which violates the GBDH assumption. □