Public Traceability in Traitor Tracing Schemes

Hervé Chabanne[1], Duong Hieu Phan[2], and David Pointcheval[2]

[1] SAGEM, Eragny, France
[2] CNRS/ENS, Computer Science Department, Paris, France
http://www.di.ens.fr/users/{phan,pointche}

Abstract. Traitor tracing schemes are of major importance for secure distribution of digital content. They indeed aim at protecting content providers from colluding users to build pirate decoders. If such a collusion happens, at least one member of the latter collusion will be detected. Several solutions have already been proposed in the literature, but the most important problem to solve remains having a very good ciphertext/plaintext rate. At Eurocrypt '02, Kiayias and Yung proposed the first scheme with such a constant rate, but still not optimal. In this paper, granted bilinear maps, we manage to improve it, and get an "almost" optimal scheme, since this rate is asymptotically 1. Furthermore, we introduce a new feature, the "public traceability", which means that the center can delegate the tracing capability to any "untrusted" person. This is not the first use of bilinear maps for traitor tracing applications, but among the previous proposals, only one has remained unbroken: we present an attack by producing an anonymous pirate decoder. We furthermore explain the flaw in their security analysis. For our scheme, we provide a complete proof, based on new computational assumptions, related to the bilinear Diffie-Hellman ones, in the standard model.

1 Introduction

The secure distribution of digital content to a set of subscribers is an important application of global networking (e.g. pay-per-view television.) There are two main types of schemes in the literature to deal with this topic: broadcast encryption schemes, which enable a center to prevent a set of users from recovering the broadcasted information; and traitor tracing schemes, which enable the center to trace users who collude to produce pirate decoders. Both types of schemes can be trivially combined by XOR'ing the results as shown in [7, 8]. There are also several works considering efficient combinations of the two attributes of *broadcast capability* and *traceability* [9, 10, 19, 17]. This paper focuses on the *traceability property.* As mentioned in the seminal paper on *traitor tracing* of Chor *et .al* [7, 8], a c-traitor tracing scheme should guarantee that:

1. either the cleartext information itself is continuously transmitted to the enemy by a traitor;
2. or any captured pirate decoder will correctly identify a traitor and will protect the innocent even if up to c traitors collude.

There is indeed no technical way to prevent a pirate from decoding and forwarding the stream to a user. But this would be rather expensive and commercially unattractive. Therefore, traitor tracing schemes deal with the traceability of pirate decoders only.

1.1 Transmission Rates

A direct solution to the traitor tracing problem is to give to each subscriber an individual key and encrypt the data separately under each key. But this is extremely inefficient because this means that the total size of the broadcast ciphertext is at least n times the size of the plaintext, where n is the number of authorized users: the ciphertext/plaintext rate is thus greater than n. The transmission rate [14] has a quite important practical impact. It actually collects three parameters: ciphertext rate, encryption-key rate and user-key rate, which are respectively the ratio of the size of ciphertext, encryption-key and user-key over the size of the plaintext (in an asymptotic way.) We thus have two main categories for traitor tracing schemes:

1. Schemes with *constant* transmission rate [14]. They are well-suited to encrypt large messages. Another interesting advantage of these schemes is the efficient black-box traceability. This means that the tracing procedure does not have to open the pirate decoder, but just to interact with it. On the other hand, the constant transmission rate is asymptotically achieved, and thus for large plaintexts only (this is due to the use of collision-secure codes.)
2. Schemes with *no constant* transmission rate [4, 2, 15]. The main advantage of these schemes is about their relatively small size of admissible plaintexts. However, the transmission rate is often linear w.r.t the maximal number of colluders. Furthermore, in these schemes, there is no efficient black-box traitor tracing. It is possible to do black-box traitor tracing [2], but it is shown that the algorithm is non-realistic because of the complexity which is larger than the binomial of n and c, where n is the number of users and c is the maximal number of colluders.

According to the context, one may use a scheme from the first category or a scheme from the second one: if one wants to distribute large messages, the first category is much more suitable, however if one simply wants to exchange a session key, which size is relatively small, the second category may be better from efficiency point of view, but the actual security can be discussed because of the inefficient black-box tracing procedure. In this paper, we further improve the transmission rate of the unique above constant transmission rate scheme [14].

1.2 Traceability

In all known traitor tracing schemes, only the center, owning some crucial private information, can execute the tracing procedure: delegation is not possible, unless the center discloses private information allowing to trace, but also to create new *anonymous* decoders, which is not reasonable.

However, such a delegation could be a quite interesting feature: if the center is the only server able to run the tracing procedure, a bottleneck may appear because of a possibly large number of pirate decoders.

This paper thus introduces a new property, called *public traceability*: the tracing procedure can be publicly done, by simply providing the tracing information, which just helps to trace, but nothing else.

1.3 Bilinear Maps

Let us now turn to the tool recently introduced in cryptographic protocols by Joux [13]: the use of some specific bilinear maps, such as the modified Weil pairing or the Tate pairing. They have already been widely used to achieve new features, such as identity-based cryptosystems [3], or to improve the efficiency of some schemes [1]. However, such particular properties could be used by adversaries too, in order to break underlying schemes such as the attacks from [20] on the traitor tracing scheme proposed in [16].

In this paper, we show these two sides of the use of bilinear maps. On the one hand, we show how the pairings can be used for improving a traitor tracing scheme, in two directions. It indeed helps to get a more efficient scheme as well as the new feature of *public traceability*. On the other hand, we show that the adversaries can also take advantage of them in some schemes: we present an attack against the only unbroken traitor tracing scheme based on pairings [20].

1.4 Contribution

At Eurocrypt '02, Kiayias and Yung [14] proposed a new traitor tracing scheme (named KY in the following) with constant transmission rates: the ciphertext rate is 3, the encryption-key rate is 4 and the user-key rate is 2.

In this paper we propose a scheme which further improves them: the ciphertext rate is reduced to 1 (asymptotically), which is optimal; the encryption-key rate is reduced to 1; and the user-key rate is kept unchanged. As already noticed, these transmission rates are considered in the multi-user setting, when the number of users is large, and when the size of the plaintext is large too. Above improvements are achieved, while still keeping the two extremely desirable properties, as in the KY scheme:

- *public-key* traitor tracing, where any third party is able to send secure messages to the set of subscribers;
- efficient *black-box* traitor tracing in which the tracing procedure can be accomplished without opening the pirate decoder.

We furthermore introduce a new quite interesting functionality: the *public traceability*. In all previous traitor tracing schemes, only the center, owning some crucial private information, could execute the tracing procedure. In our scheme, the center can publish some information in such a way that every one can do the tracing procedure, at least the interactive part with the pirate decoder.

As already said, pairings are of great help to achieve this goal. But care is required. To the best of our knowledge, only one such a scheme based on pairings has remained unbroken: the scheme proposed by To, Safavi-Naini and Zhang [20] (named TSZ in the following). In this paper we show an attack on the TSZ scheme: we exhibit a way to produce an anonymous pirate decoder, while they provided a security proof. We thereafter explain where is the flaw in their tracing algorithm.

2 TSZ: the To, Safavi-Naini and Zhang's Scheme

Mitsunari, Sakai and Kasahara [16] proposed the first traitor tracing scheme using the bilinear maps. One year later, To, Safavi-Naini and Zhang [20] presented an attack and tried to repair it. Unfortunately, this modification is not correct either. Let us first review it, then we present an attack. This scheme and the attack will help to understand later our new construction which is a combination of the TSZ scheme and the KY scheme, taking advantage of the best of each.

2.1 Description of the Scheme

The TSZ scheme uses a bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$, where $\mathcal{G}_1, \mathcal{G}_2$ are groups of prime order q (see section 3 for a brief review.)

Initialization: two arbitrary random generators $P, Q \in \mathcal{G}_1$ and a unitary polynomial with coefficients in $\mathbb{Z}_q$ of degree $2k-1$:

$$f(x) = a_0 + a_1 x + \ldots + a_{2k-2} x^{2k-2} + x^{2k-1}.$$

Let $Q_0 = a_0 Q$, $Q_1 = a_1 Q$, $\ldots$, $Q_{2k-2} = a_{2k-2} Q$ and $g = \hat{e}(P, Q) \in \mathcal{G}_2$.

Private key of the center: the generator P, and the polynomial f.

Encryption key: the tuple $(g, Q, Q_0, Q_1, \ldots, Q_{2k-2})$.

User key (for user u): $K_u = f(u)^{-1} P$.

Encryption Algorithm: one generates a random $r \in \mathbb{Z}_q$, then the session key $s \in \mathcal{G}_2$ is encrypted into: $c = (sg^r, rQ, rQ_0, \ldots, rQ_{2k-2})$.

Decryption Algorithm: user u first computes g^r, granted K_u, and then recovers s. Indeed, $g^r = \hat{e}(K_u, rQ_0) \times \ldots \times \hat{e}(u^{2k-2} K_u, rQ_{2k-2}) \times \hat{e}(u^{2k-1} K_u, rQ)$.

2.2 Attack

In [20], authors showed that nobody can build an anonymous decoder, even a collusion of registered users. Here, we explain how a unique user can build such an anonymous decoder: user u chooses random elements $z_0, z_1, \ldots, z_{2k-2}$ in $\mathbb{Z}_q$ and produces the following decoder:

- $X_i = u^i K_u + z_i Q$, for i from 0 to $2k-2$.
- X_{2k-1} is determined by the relation:

$$X_{2k-1} = u^{2k-1} K_u - (z_0 Q_0 + z_1 Q_1 + \ldots + z_{2k-2} Q_{2k-2}). \qquad (1)$$

User u can then publish $X_0, X_1, \ldots, X_{2k-1}$, which provides everyone with the ability of recovering $g^r = \hat{e}(X_0, rQ_0) \times \ldots \times \hat{e}(X_{2k-2}, rQ_{2k-2}) \times \hat{e}(X_{2k-1}, rQ)$.

2.3 Flaw in the Security Analysis

While authors provided a tracing procedure, we now show that our above decoder, which only uses $X_0, \ldots, X_{2k-2}, X_{2k-1}$, cannot trace back user u. First, $X_0, \ldots, X_{2k-2}, X_{2k-1}$ satisfy the following relation:

$$\begin{aligned} \sum_0^{2k-1} a_i X_i &= \left(\sum_0^{2k-2} a_i u^i \right) K_u + \left(\sum_0^{2k-2} a_i z_i \right) Q + u^{2k-1} K_u - \left(\sum_0^{2k-2} z_i a_i \right) Q \\ &= f(u) K_u = P. \end{aligned} \tag{2}$$

Remark also that for each user, and all i ($i = 0, \ldots, 2k-2$), the application, from $\mathbb{Z}_q$ to $\mathcal{G}_1$, which maps z_i to X_i, is a bijection. Therefore, instead of choosing $z_0, z_1, \ldots, z_{2k-2}$, one can randomly choose $X_0, \ldots, X_{2k-2}$ in $\mathcal{G}_1$, which uniquely defines the tuple $(z_0, z_1, \ldots, z_{2k-2})$. Thereafter, X_{2k-1} is also uniquely determined by the relation (1). It also satisfies the relation (2). Hence, one can formally define it from the latter relation: it thus clearly does not depend on u.

As a consequence, one easily sees that all the users would produce the same set of pirate decoders, with parameters $(X_0, X_1, \ldots, X_{2k-2}, X_{2k-1})$, so that $X_0, X_1, \ldots, X_{2k-2}$ are randomly chosen in $\mathcal{G}_1^{2k-1}$, while X_{2k-1} is defined according to the relation (2).

Note that this attack is quite different from the one in [20]. Our pirate decoder indeed combines informations of the user-key, together with the *public information* of the system. The latter part points out the flaw in the tracing algorithm from [20], which works as follows: for a suspect set of users $A = \{u_1, \ldots, u_t\}$ (whose size is up to k), they construct another polynomial $f'(x) = f(x) + \alpha \times (x - u_1) \times \ldots \times (x - u_t)$. For any user in the set A, his key in the scheme using f (named $\mathsf{Scheme}(f)$) and the one in the scheme using f' (named $\mathsf{Scheme}(f')$) are identical. For this reason, they claimed that if the colluders are in the set A, then any pirate decoder produced by them in $\mathsf{Scheme}(f)$ is also a pirate decoder in $\mathsf{Scheme}(f')$. Accordingly, this decoder will decrypt a ciphertext in $\mathsf{Scheme}(f')$ as it would be in $\mathsf{Scheme}(f)$. Therefore, by sending a decryption query to the decoder, the center can easily detect whether the set of colluders is included in A or not.

Unfortunately, their argument is not correct. If the construction of the pirate decoder depends only on the user-keys of the colluders, their tracing algorithm works well. But if the construction depends on the public information too (which are of course available to the colluders), the tracing procedure fails, as shown above.

3 Bilinear Maps and Computational Assumptions

3.1 Bilinear Maps

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two groups of order q, for some large prime q. We use in our system a bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$, which must satisfy the following properties:

Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathcal{G}_1$ and all $a, b \in \mathbb{Z}_q$;

Non-degenerated: The map does not send all pairs in $\mathcal{G}_1 \times \mathcal{G}_1$ to the unit in $\mathcal{G}_2$;

Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any elements $P, Q \in \mathcal{G}_1$.

A bilinear map satisfying the three above properties is said to be an *admissible bilinear map*. Throughout the paper we view $\mathcal{G}_1$ as an additive group and $\mathcal{G}_2$ as a multiplicative group. Remark that since $\mathcal{G}_1, \mathcal{G}_2$ are groups of prime order and $\hat{e}$ is non-degenerated, if P is a generator of $\mathcal{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathcal{G}_2$.

Example: The modified Weil pairing or the Tate pairing can be used to construct an admissible bilinear map that satisfies the three above properties.

3.2 Computational Assumptions

In this section, we review some well-known problems such as the computational bilinear Diffie-Hellman problems. We also propose new problems, we believe to be hard to solve. Relations claimed in propositions are provided in the full version [6]. They will be used in the next sections, in the security analysis of our scheme.

Classical Assumptions and Variants. We first review the most classical problems in $\mathcal{G}_1$.

CDH – the computational Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP) for some $a, b \in \mathbb{Z}_q^\star$, output abP.

CBDH^1 – the computational bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP, cP) for some $a, b, c \in \mathbb{Z}_q^\star$, output $abcP$.

DBDH^1 – the decisional bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP, cP, U) for some $a, b, c \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_1$, output yes if $U = abcP$ and no otherwise.

We now introduce modified versions of the two above Bilinear Diffie-Hellman problems. They are actually particular cases, where $b = c$. We then provide some relations between them and the usual CDH problem.

$\mathsf{CBDH}^1\text{-}\mathsf{M}$ – the modified computational bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given (P, aP, bP) for some $a, b \in \mathbb{Z}_q^\star$, output ab^2P.

$\mathsf{DBDH}^1\text{-}\mathsf{M}$ – the modified decisional bilinear Diffie-Hellman problem in $\mathcal{G}_1$:
Given(P, aP, bP, U) for some $a, b \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_1$, output yes if $U = ab^2P$ and no otherwise.

Proposition 1. *The* $\mathsf{CBDH}^1\text{-}\mathsf{M}$ *problem is at least as hard as the* CBDH^1 *problem, which is at least as hard as the usual* CDH *problem:*

$$(\mathsf{Succ}_{\mathcal{G}_1}^{\mathsf{CBDH}^1\text{-}\mathsf{M}}(t))^2 \leq \mathsf{Succ}_{\mathcal{G}_1}^{\mathsf{CBDH}^1}(t) \leq \mathsf{Succ}_{\mathcal{G}_1}^{\mathsf{CDH}}(t).$$

Pairing-Based Problems. We now review the bilinear Diffie-Hellman problems, all in $\mathcal{G}_1$ and $\mathcal{G}_2$, with the admissible map $\hat{e}$ (we thus omit them in the notation.)

CBDH^2 – the computational bilinear Diffie-Hellman problem:
Given (P, aP, bP, cP) for some $a, b, c \in \mathbb{Z}_q^\star$, output g^{abc}, where $g = \hat{e}(P, P)$.

DBDH^2 – the decisional bilinear Diffie-Hellman problem:
Given (P, aP, bP, cP, Z) for some $a, b, c \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_2$, output yes if $Z = g^{abc}$ and no otherwise, where $g = \hat{e}(P, P)$.

$\mathsf{CBDH}^2\text{-}\mathsf{E}$ – the extended computational bilinear Diffie-Hellman problem:
Given (P, aP, bP, cP, ab^2P) for some $a, b, c \in \mathbb{Z}_q^\star$, output g^{cb^2}, where $g = \hat{e}(P, P)$.

$\mathsf{DBDH}^2\text{-}\mathsf{E}$ – the extended decisional bilinear Diffie-Hellman problem:
Given $(P, aP, bP, cP, ab^2P, Z)$ for some $a, b, c \in \mathbb{Z}_q^\star$ and $U \in \mathcal{G}_2$, output yes if $Z = g^{cb^2}$ and no otherwise, where $g = \hat{e}(P, P)$.

We furthermore introduce a slight variant of the CBDH^2, in order to get more confidence in the above $\mathsf{CBDH}^2\text{-}\mathsf{E}$ problem:

$\mathsf{CBDH}^2\text{-}\mathsf{V}$ – a variation of the computational bilinear Diffie-Hellman problem:
Given $(P, aP, bP, cP, a(a^2 - b^2)P, b(a^2 - b^2)P)$ for some $a, b, c \in \mathbb{Z}_q^\star$, output g^{abc}, where $g = \hat{e}(P, P)$.

Proposition 2. *The* $\mathsf{CBDH}^2\text{-}\mathsf{E}$ *problem is at least as hard as the* $\mathsf{CBDH}^2\text{-}\mathsf{V}$ *problem:*

$$(\mathsf{Succ}^{\mathsf{CBDH}^2\text{-}\mathsf{E}}_{\hat{e},\mathcal{G}_1,\mathcal{G}_2}(t))^2 \leq \mathsf{Succ}^{\mathsf{CBDH}^2\text{-}\mathsf{V}}_{\hat{e},\mathcal{G}_1,\mathcal{G}_2}(t).$$

Mixed Problems. Let us now introduce new problems which involve elements from $\mathcal{G}_1$ and $\mathcal{G}_2$, still with the admissible map $\hat{e}$ (we thus omit them in the notation.)

MCDH – the mixed computational Diffie-Hellman problem:
Given (P, aP, a^2P, g^b) for some $a, b \in \mathbb{Z}_q^\star$, where $g = \hat{e}(P, P)$, output g^{ba^2}.

MDDH – the mixed decisional Diffie-Hellman problem:
Given (P, aP, a^2P, g^b, Z) for some $a, b \in \mathbb{Z}_q^\star$ and $Z \in \mathcal{G}_2$, where $g = \hat{e}(P, P)$, output yes if $Z = g^{ba^2}$ and no otherwise.

4 The Basic Building Block: The Two-User Case

4.1 The Assumptions for our Scheme

We have introduced several new problems, which will simplify the security analysis of our proposal. Let us sum up which assumptions will be really needed, according to the security level.

Traitor Tracing. Let us first consider the semantic security of the encryption scheme. In the random-oracle model, the security will hold under the MCDH assumption. In the standard model, the security relies on the stronger MDDH assumption. We believe these are reasonable assumptions.

About the traitor-tracing functionality, the non-incrimination relies on the CDH assumption, while the traceability of colluders is guaranteed under the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ assumption.

As a consequence, our scheme will achieve the classical security notions of traitor-tracing under the MDDH and $\mathsf{DBDH}^1\text{-}\mathsf{M}$ assumptions.

Public Traceability. Our scheme will provide the new and interesting property of *public traceability.* It however requires stronger assumptions, since more information is available to the adversary (since the tracing capability can be provided to a bad guy.)

About the semantic security of the encryption scheme encryption, in the random-oracle model, the $\mathsf{CBDH}^2\text{-}\mathsf{E}$ assumption is required. The latter is in fact a quite minor extension of the classical CBDH^2 assumption (see proposition 2.) In the standard model, the security relies on the $\mathsf{DBDH}^2\text{-}\mathsf{E}$ assumption. Again, we believe this is a reasonable assumption.

Considering the properties of traitor-tracing, the non-incrimination is captured by the tracing of colluders, which is again proven under the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ assumption.

Conclusion. Finally, our scheme, with public traceability, will essentially require the three new assumptions MDDH for the security of encryption, $\mathsf{DBDH}^1\text{-}\mathsf{M}$ for the traitor-tracing property, and the $\mathsf{DBDH}^2\text{-}\mathsf{E}$ for the public traceability.

4.2 Kiayias-Yung's Scheme

Our construction of 2-user traitor tracing scheme is based on the Kiayias and Yung's scheme [14], which can be seen as a special case of the Boneh and Franklin's scheme [2]. Let us thus first review the KY scheme.

Setup: Given a security parameter $\kappa \in \mathbb{Z}$, the algorithm works as follows:
- **Step 1:** Generate a κ-bit prime q and a group $\mathcal{G}$ of order q. Choose an arbitrary generator $g \in \mathcal{G}$.
- **Step 2:** Pick random elements $a, z \in \mathbb{Z}_q^\star$, and set $Q = g^a$, $Z = g^z$.

Private key of the center: the pair (a, z).

Encryption key: the tuple $\mathsf{pk} = (g, Q, Z)$.

User key: user u_b (for $b \in \{0, 1\}$, since we focus on the two-user case) is associated to a "representation" $k_b = (\alpha_b, \beta_b)$ of g^z with respect to the basis (g, g^a), i.e, the authority selects two vectors (α_0, β_0) and (α_1, β_1) in $\mathbb{Z}_q^2$ so that $\alpha_b + a\beta_b = z \bmod q$ for both $b \in \{0, 1\}$. The two vectors are chosen so that they are linearly independent. The set of all possible keys is

$$\mathcal{K}_{\mathsf{pk}} = \{(\alpha, \beta) | \alpha + a\beta = z \bmod q\}.$$

Encryption Algorithm: The encryption algorithm generates a random $k \in \mathbb{Z}_q$ and outputs a ciphertext (c_1, c_2, d) into $\mathcal{G}^3$: on a plaintext m, assumed to be in the group $\mathcal{G}$, the center computes $C = (c_1 = g^k, c_2 = Q^k, d = m \times Z^k)$. We say that a triple $(c_1, c_2, d) \in \mathcal{G}^3$ is a valid ciphertext if there exits $k \in \mathbb{Z}_q$ such that $c_1 = g^k$ and $c_2 = Q^k$. Otherwise, the ciphertext is invalid.

Decryption Algorithm: On a ciphertext (c_1, c_2, d), user u_b computes:

$$Z^k = c_1^\alpha \times c_2^\beta \quad \text{and} \quad m = d/Z^k.$$

4.3 Our Construction

We now show how we can use bilinear maps in order to improve this scheme. More precisely, we introduce the notion of public-key traitor tracing with *proxy quantity*. Contrarily to usual public-key traitor tracing schemes, the authority generates for each user a key along with a corresponding *proxy quantity*. The authority keeps in his hands the user's key and gives only to the user the proxy quantity which is enough for decryption. The user's key will be later used for tracing.

Setup: Given a security parameter $\kappa \in \mathbb{Z}$, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathcal{G}_1$ and $\mathcal{G}_2$ of order q, and an admissible bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$. Choose an arbitrary generator $P \in \mathcal{G}_1$ and set $g = \hat{e}(P, P)$ which is a generator of group $\mathcal{G}_2$.

Step 2: Pick random elements $a, z \in \mathbb{Z}_q^\star$, and set $Q = aP$, $Z = g^z$.

Step 3: Choose a function $H : \mathcal{G}_1 \to \mathcal{M}$. The security analysis will view H as either a random oracle or a function in a universal hash function family (using the leftover-hash-lemma [11, 12]).

The message space is $\mathcal{M} = \{0,1\}^\kappa$. The ciphertext space is $\mathcal{G}_1^\star \times \mathcal{G}_1^\star \times \{0,1\}^\kappa$. The system parameters are $\mathsf{params} = (q, \mathcal{G}_1, \mathcal{G}_2, \hat{e}, P, H)$.

Private key of the center: the pair (a, z).

Encryption key: the tuple $\mathsf{pk} = (g, Q, Z)$.

User key: user u_b (for $b \in \{0,1\}$) is associated to a "representation" $k_b = (\alpha_b, \beta_b)$ of g^z with respect to the base (g, g^a). The set of all possible keys is

$$\mathcal{K}_{\mathsf{pk}} = \{(\alpha, \beta) | \alpha + a\beta = z \bmod q\}.$$

Remark that the authority generates these keys for each user but *does not give them* to the users. Each user is just given a proxy quantity, as described below.

Proxy Quantity: user u_b (for $b \in \{0,1\}$) receives a proxy quantity $\Pi(k_b) = (\alpha_b, \pi_b = \beta_b P)$. The set of all possible proxy quantities is

$$\Pi_{\mathsf{pk}} = \{(\alpha, \pi = \beta P) | (\alpha, \beta) \in \mathcal{K}_{\mathsf{pk}}\}.$$

Encryption Algorithm: The encryption algorithm generates a random $k \in \mathbb{Z}_q$ and outputs a ciphertext (c_1, c_2, d) into $\mathcal{G}_1 \times \mathcal{G}_1 \times \mathcal{M}$: on a plaintext $m \in \mathcal{M}$, the center computes $C = (c_1 = kP, c_2 = k^2 Q, d = m \oplus H(Z^{k^2}))$. We say that $(c_1, c_2, d) \in \mathcal{G}_1 \times \mathcal{G}_1 \times \mathcal{M}$ is a valid ciphertext if there exits $k \in \mathbb{Z}_q$ such that $c_1 = kP$ and $c_2 = k^2 Q$. Otherwise, the ciphertext is invalid.

Decryption Algorithm: On a ciphertext (c_1, c_2, d), user u_b computes, granted his proxy quantity $\Pi(k_b) = (\alpha_b, \pi_b)$,

$$Z^{k^2} = \hat{e}(\alpha_b c_1, c_1) \cdot \hat{e}(\pi_b, c_2) \quad \text{and} \quad m = d \oplus H(Z^{k^2}).$$

4.4 Rationale

First, one can wonder why we do not encrypt the message by $c_1 = kP$, $c_2 = kQ$ and $d = m \oplus H(Z^k)$, while each user would receive the key $k_b = (\alpha_b, \beta_b)$ (so that $\alpha_b + a\beta_b = z \bmod q$). Such scheme would thus be a simple and natural variation of the KY scheme using bilinear maps. However, as in our above attack against the TSZ scheme, the adversary could take advantage of the bilinear property to combine the secret key and the public information. Actually, any adversary, although it could not produce a new key, could produce and distribute an anonymous decoder $(X = \alpha P - uQ, Y = \beta P + uP)$, in which u could be randomly chosen in $\mathbb{Z}_q$. Then, everyone could recover $Z^k = \hat{e}(X, c_1) \cdot \hat{e}(Y, c_2)$. Because of the random choice of u, the authority cannot trace back the traitor.

In our scheme, we prove that such an adversary cannot exist: users do not have keys of the form (α, β), but proxy quantities only, of the form $(\alpha, \beta P)$. As a consequence, even if two users collude to produce another key (by a linear combination of their keys), they cannot learn the secret key (a, z). We will see that this is crucial to improve the result in the multi-user case.

4.5 Security of the Encryption Scheme

Before considering security properties specific to the traitor tracing functionality, let us first study the encryption scheme. Actually, if we consider the function H as a random oracle, the semantic security of the encryption can be proved under the MCDH problem. If we consider that the function H is randomly chosen in a universal hash function family [11, 12], the semantic security of the encryption is proved under the MDDH problem. The proofs of the following theorems can be found in the full version [6].

Theorem 3. *Let H be seen as a random oracle. The above scheme is semantically secure under the* MCDH *problem.*

Theorem 4. *Let H be a function randomly chosen in a universal hash function family. The above scheme is semantically secure under the* MDDH *problem.*

4.6 Non-Incrimination

The main goal of a traitor tracing scheme is to be able to trace a pirate. But a pirate could try to incriminate another user. E.g., using his private information, a pirate could try to produce another proxy quantity and distribute it. We show that this scenario cannot happen. The proof can be found in the appendix.

Theorem 5. *Given the encryption key and a proxy quantity $(\alpha, \pi) \in \Pi_{\mathsf{pk}}$, it is computationally infeasible to construct another proxy quantity in Π_{pk} under the* CDH *problem.*

4.7 Black-Box Traitor Tracing

For practical reasons, it is important not to have to open the pirate decoder in order to trace back the pirate. We thus show that our scheme is black-box traitor tracing against a collusion of the 2 users under the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ problem, by constructing a tracing algorithm. For this security result, we assume that the hash function H is a function randomly chosen in a universal hash function family. The proof can be found in the full version [6], since it is a simpler case than the proof of the Theorem 9 (provided in the appendix.)

Theorem 6. *Let us assume that, given the encryption key* pk *and a proxy quantity* $(\alpha, \pi = \beta P) \in \Pi_{\mathsf{pk}}$*, the adversary* $\mathcal{A}$ *produces a decryption simulator* $\mathcal{S}$ *that decrypts valid ciphertexts, but when given a "randomized" ciphertext of the form* (kP, ak'^2P, d) *with* $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q, d \stackrel{R}{\leftarrow} \mathcal{M}$*, it outputs a value different from* $d \oplus H(g^{\alpha k^2 + a\beta k'^2})$ *with probability* ε*. Then the* $\mathsf{DBDH}^1\text{-}\mathsf{M}$ *problem can be solved with an advantage* $\varepsilon/2$*.*

Intuitively, the above theorem shows that a "randomized" and thus invalid ciphertext cannot be distinguished from a regular and valid ciphertext. Therefore, given a black-box access to a decryption simulator $\mathcal{S}$ constructed by one of two users, one can always decide which one of them has built it: one randomly chooses $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$ (we suppose that $k \neq k'$), and sets $u_0 = \alpha_0 k^2 + a\beta_0 k'^2$ and $u_1 = \alpha_1 k^2 + a\beta_1 k'^2$. With high probability (greater than $1 - 2/q$), u_0 is different from u_1, which is thus assumed in the following. One then submits the randomized invalid ciphertext (kP, ak'^2P, d). If the output of $\mathcal{S}$ is d/g^{u_0} then one claims that u_0 is the traitor. If the output is d/g^{u_1}, then u_1 is blamed. If the output is none of these two values, one concludes that the two users colluded. Hence the following corollary.

Corollary 7. *The above scheme is black-box traitor tracing against active adversaries.*

4.8 Public Traceability

Let us now turn to the additional and quite interesting property: in order to execute the black-box traitor tracing procedure, the two user-keys (α_0, β_0) and (α_1, β_1) are used. However, the proxy quantities would be enough, and even less: $(\alpha_0 P, \beta_0 P)$ and $(\alpha_1 P, \beta_1 P)$ are sufficient. From $k, k' \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$, one does not really need u_0, u_1, but just g^{u_0} and g^{u_1}:

$$\begin{aligned} g^{u_0} &= \hat{e}(\alpha_0 P, k^2 P) \times \hat{e}(Q, k'^2(\beta_0 P)); \\ g^{u_1} &= \hat{e}(\alpha_1 P, k^2 P) \times \hat{e}(Q, k'^2(\beta_1 P)). \end{aligned}$$

This is a quite new and interesting property: one can split the roles of the center. Moreover, the tracing capability can be delegated to several servers in order to speed up the tracing. This delegation does not require any trust in these servers, since the given information does not leak the private key (a, z), nor even any information to build a decoder (under an additional computational assumption). Furthermore, one can thereafter check whether the incriminated people are the pirates or not.

We now formally state the above security properties in the following theorems whose proofs can be found in the appendix.

Theorem 8. *Let us assume that the tracing information is public, then the encryption scheme is semantically secure: in the random-oracle model, the security relies on the* $\mathsf{CBDH}^2\text{-}\mathsf{E}$ *assumption, while the standard model requires the* $\mathsf{DBDH}^2\text{-}\mathsf{E}$ *assumption.*

Theorem 9. *Let us assume that* $\mathcal{A}$ *is an algorithm which, given the encryption key* pk*, one proxy quantity* $(\alpha, \pi = \beta P)$ *(among the two* $(\alpha_0, \pi_0 = \beta_0 P)$ *and* $(\alpha_1, \pi_1 = \beta_1 P)$ *provided by the center), and the public tracing information* $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$*, can produce a decryption simulator* $\mathcal{S}$ *that decrypts valid ciphertexts, but when given a "randomized" ciphertext of the form* $(kP, ak'^2 P, d)$ *with* $k, k' \xleftarrow{R} \mathbb{Z}_q$*,* $d \xleftarrow{R} \mathcal{M}$*,* $\mathcal{S}$ *outputs a value different than* $d \oplus H(g^{\alpha_0 k^2 + a\beta_0 k'^2})$ *with probability* ε*. Then the* $\mathsf{DBDH}^1\text{-}\mathsf{M}$ *problem can be solved with advantage* $\varepsilon/2$*.*

5 The Multi-User Case

5.1 Description

Let $C = \{\omega_1, \ldots, \omega_N\}$ be an $(N, c, \ell, \varepsilon)$-collusion-secure code over the alphabet $\{0, 1\}$ with ℓ-long codewords, that allows collusions of up to c users and has a tracing algorithm that succeeds with probability $1 - \varepsilon$ (see [4] for more details). The multi-user case (ℓ-key system) is simply ℓ-instantiations of the 2-user public-key 1-traitor tracing scheme with proxy quantities. We indeed build such an ℓ-key system using an $(N, c, \ell, \varepsilon)$-collusion-secure code C as a combination of ℓ 2-user systems $S_1, S_2, \ldots, S_\ell$:

Setup: Given the security parameters k, c and ε:

- **Step 1:** Generate a k-bit prime q, two groups $\mathcal{G}_1$, $\mathcal{G}_2$ of order q, and an admissible bilinear map $\hat{e} : \mathcal{G}_1 \times \mathcal{G}_1 \to \mathcal{G}_2$. Choose an arbitrary generator $P \in \mathcal{G}_1$.
- **Step 2:** Generate an $(N, c, \ell, \varepsilon)$-collusion-secure code $C = \{\omega_1, .., \omega_N\}$.
- **Step 3:** Pick random elements $a, z_j \in \mathbb{Z}_q^\star$, and set $Q = aP$, $Z_j = g^{z_j}$, for $j = 1, \ldots, \ell$.
- **Step 4:** Choose a function $H : \mathcal{G}_1 \to \mathcal{M}$.

The system parameters are $\mathsf{params} = (q, \mathcal{G}_1, \mathcal{G}_2, \hat{e}, P, H)$. These parameters are common for all 2-user systems $S_1, S_2, \ldots, S_\ell$.

Private key of the center: the element a, and the tuple $(z_j)_{j=1,\ldots,\ell}$.

Encryption key: this is the combination of the encryption keys from the ℓ 2-user schemes: $\mathsf{pk} = (g, Q, \{Z_j = g^{z_j}\}_{j=1,\ldots,\ell})$.

User key: user u_i (for $i \in \mathbb{Z}_N$) is associated to a codeword ω_i in C and the corresponding "representation" $(\alpha_{\omega_{i,j},j}, \beta_{\omega_{i,j},j})$ of g^{z_j} with respect to the basis (g, g^a), where $\omega_{i,j}$ is the j-th bit of the codeword ω_i. Recall that $(\alpha_{b,j}, \beta_{b,j})$ is a "representation"of g^{z_j} with respect to the base (g, g^a). Again, this user key is *not* given to the user, but only the proxy quantity.

Proxy Quantity: user u_i (for $i \in \mathbb{Z}_N$) is given the proxy quantity $\Pi_i = (\Pi_{\omega_{i,1},1}, \ldots, \Pi_{\omega_{i,l},\ell})$. More precisely, for $j = 1, \ldots, \ell$,

$$\Pi_{\omega_{i,j},j} = (\alpha_{\omega_{i,j},j}, \pi_{\omega_{i,j},j} = \beta_{\omega_{i,j},j} P).$$

Encryption algorithm: The plaintext space of the ℓ-key system is $\mathcal{M}^\ell$. On input $(m_1, \ldots, m_\ell)$, the encryption algorithm uses a random $k \in \mathbb{Z}_q$ and outputs the ciphertext $(c_1, c_2, d_1, \ldots, d_\ell)$ into $\mathcal{G}_1 \times \mathcal{G}_1 \times \mathcal{G}_2^\ell$, where: $c_1 = k \times P$, $c_2 = k^2 \times aP$ and $d_j = m_j \oplus H(Z_j^{k^2})$.

Decryption Algorithm: On the ciphertext $(c_1, c_2, d_1, \ldots, d_\ell)$, user u_i computes, granted his proxy quantity, $Z_j^{k^2} = \hat{e}(\alpha_{\omega_{i,j},j} c_1, c_1) \times \hat{e}(\pi_{\omega_{i,j},j}, c_2)$ and then $m_j = d_j \oplus H(Z_j^{k^2})$.

For the security analysis, one could use the following assumption, from [14]: the *threshold assumption* says that a pirate-decoder that just returns correctly a fraction p of a plaintext of length λ where $1 - p$ is a non-negligible function in λ, is useless. However, as already mentioned in [14], by employing an all-or-nothing transform [18, 5], this assumption is not necessary.

Proposition 10. *The collusion of the users in the $(\ell - 1)$ 2-user systems of ℓ 2-user systems does not affect the security of the remained 2-user system.*

This proposition which proof can be found in the full version [6], combined with the fact that C is an $(N, c, \ell, \varepsilon)$-collusion-secure code, leads to following corollary:

Corollary 11. *The above scheme is a N-user, c-traitor tracing scheme.*

About the public traceability, with the public information, anybody can recover the codeword associated to the pirate decoder, the interactive and thus costly phasis. However, classical collusion-secure codes do not allow to publicly trace back to a guilty, but this is an off-line prodecure, which still must be performed by a trusted authority.

5.2 Comparison with the Kiayias-Yung's Scheme

In the KY scheme, the ciphertext rate is 3, while ours is asymptotically 1. One could wonder why we could use the above construction, while they could not.

The reason is that in our 2-user scheme, even the collusion of the 2 users does not leak any information about a. In the KY 2-user scheme, such a collusion immediately reveals a: in the multi-user case, if one uses the same a for the ℓ 2-user schemes, the collusion of two users could leak this value a and then all the values z_i, which would easily lead to an anonymous pirate decoder. As a consequence, they have to use distinct a's in each 2-user scheme instance, while in our scheme, a common a is possible.

6 Conclusion

We thus proposed a scheme which improves the Kiayias and Yung's scheme in various ways: first, the transmission rates are reduced near optimality; and we introduce the quite interesting functionality of *public traceability*. The full feature of public traceability in the multi-user case, which would lead to a guilty, is however an open problem.

Acknowledgement

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.

References

1. D. Boneh and X. Boyen. Short signatures without random oracles. In *Adv. in Cryptology – Proceedings of Eurocrypt 2004*, volume LNCS 3152, pages 56–73. Springer-Verlag, 2004.
2. D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In M.Wiener, editor, *Adv. in Cryptology – Proceedings of Crypto 1999*, volume LNCS 1666, pages 338–353. Springer-Verlag, 1999.
3. D. Boneh and M. Franklin. Identity-based Encryption from the Weil Pairing. In J. Kilian, editor, *Adv. in Cryptology – Proceedings of Crypto '01*, LNCS 2139, pages 213–229. Springer-Verlag, Berlin, 2001.
4. D. Boneh and J. Shaw. Collusion secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.
5. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-Or-Nothing Transforms. In *Eurocrypt '00*, LNCS 1807, pages 453–469. Springer-Verlag, Berlin, 2000.
6. H. Chabanne, D. H. Phan, and D. Pointcheval. Public Traceability in Traitor Tracing Schemes. In *Eurocrypt '05*, LNCS. Springer-Verlag, Berlin, 2005. Full version available from `http://www.di.ens.fr/users/pointche`.
7. B. Chor, A. Fiat and M. Naor. Tracing traitor. In Y. Desmedt, editor, *Adv. in Cryptology – Proceedings of Crypto 1994*, volume LNCS 839, pages 257–270. Springer-Verlag, 1994.
8. B. Chor, A. Fiat, M. Naor and B. Pinkas. Tracing traitor. *IEEE Transactions on Information Theory*, 46(3): 893–910, 2000.
9. Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Y.G.Desmedt, editor, *Proceedings of PKC 2003*, volume LNCS 2567, pages 100–115. Springer-Verlag, 2003.
10. E. Gagni, J. Staddon, and Y.L. Yin. Efficient methods for integrating traceability and broadcast encryption. In M.Wiener, editor, *Adv. in Cryptology – Proceedings of Crypto 1999*, volume LNCS 1666, pages 372–387. Springer-Verlag, 1999.
11. J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A Pseudorandom Generator from any One-Way Function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.
12. I. Impagliazzo, L. Levin, and M. Luby. Pseudo-Random Generation from One-Way Functions. In *Proc. of the 21st STOC*, pages 12–24. ACM Press, New York, 1989.

13. A. Joux. A One-Round Protocol for Tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium (ANTS IV)*, LNCS 1838, pages 385–394. Springer-Verlag, Berlin, 2000.
14. A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In L. Knudsen, editor, *Adv. in Cryptology – Proceedings of Eurocrypt 2002*, volume LNCS 2332, pages 450–465. Springer-Verlag, 2002.
15. A. Kiayias and M. Yung. Breaking and repairing asymmetric public-key traitor tracing. In J. Feigenbaum, editor, *ACM Workshop in Digital Rights Management – DRM 2002*, volume LNCS 2696, pages 32–50. Springer, 2003.
16. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing scheme. *IEICE Trans. Fundamentals*, E85-A(2), 2002.
17. M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Proc. of Financial Crypto '2000*, volume LNCS 1692, pages 1–20. Springer-Verlag, 2000.
18. R. Rivest. All-or-Nothing Encryption and the Package Transform. In *Proc. of the 4th FSE*, LNCS 1267. Springer-Verlag, Berlin, 1997.
19. V.D. To and R. Safavi-Naini. Linear code implies public-key traitor tracing with revocation. In H. Wang, editor, *Proceedings of ACISP 2003*, volume LNCS 3108, pages 24–35. Springer-Verlag, 2003.
20. V.D. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. In *Proceedings of the 2003 ACM Workshop on Digital Rights Management*, pages 67–76, 2003.

A Proof for the Non-Incrimination and Traitor-Tracing

A.1 Proof of the Theorem 5

Let us assume that, given a proxy quantity $(\alpha, \pi) \in \mathit{\Pi}_{\mathsf{pk}}$, $\mathcal{A}$ can construct another proxy quantity. We then construct an algorithm $\mathcal{B}$ that solves the inverse Diffie-Hellman problem, which is well-know to be equivalent to the CDH problem. Algorithm $\mathcal{B}$ is given as input a random instance $(P, A = aP)$ of the inverse Diffie-Hellman problem. Let $B = a^{-1}P$ be the solution, that $\mathcal{B}$ finds as follows:

Setup: $\mathcal{B}$ randomly chooses $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$ and $\pi \stackrel{R}{\leftarrow} \mathcal{G}_1$), and then computes $Z = \hat{e}(P, \alpha P)\hat{e}(A, \pi)$. Finally, $\mathcal{B}$ sets the public key $\mathsf{pk} = (g, A, Z)$. It then sends pk to $\mathcal{A}$ as well as a proxy quantity (α, π).
Attack: $\mathcal{A}$ outputs another proxy $(\tilde{\alpha}, \tilde{\pi})$.
Break: $\mathcal{B}$ computes $c = \tilde{\alpha} - \alpha$ and outputs $B = (c^{-1} \bmod q)(\pi - \tilde{\pi})$.

We see that, since $(\tilde{\alpha}, \tilde{\pi})$ is a new proxy, for any ciphertext (c_1, c_2, d),

$$\begin{aligned} & \hat{e}(\alpha c_1, c_1) \times \hat{e}(\pi, c_2) = \hat{e}(\tilde{\alpha} c_1, c_1) \times \hat{e}(\tilde{\pi}, c_2) \\ \iff & \hat{e}(\alpha kP, kP) \times \hat{e}(\pi, ak^2P) = \hat{e}(\tilde{\alpha} kP, kP) \times \hat{e}(\tilde{\pi}, ak^2P) \\ \iff & \hat{e}(\alpha P, P) \times \hat{e}(\pi, aP) = \hat{e}(\tilde{\alpha} P, P) \times \hat{e}(\tilde{\pi}, aP) \\ \iff & \hat{e}((\tilde{\alpha} - \alpha)P, P) = \hat{e}(\pi - \tilde{\pi}, aP). \end{aligned}$$

From the fact that $(\alpha, \pi), (\tilde{\alpha}, \tilde{\pi}) \in \mathit{\Pi}_{\mathsf{pk}}$, we get that π and $\tilde{\pi}$ are in the group $\mathcal{G}_1$ generated by P. Therefore, $\pi - \tilde{\pi} = bP$ for some b. We then have $c = \tilde{\alpha} - \alpha = ab$ and thus:

$$B = (c^{-1} \bmod q)(\pi - \tilde{\pi}) = (ab)^{-1} \times bP = a^{-1}P.$$

□

B Proofs for the Public Traceability

B.1 Proof of the Theorem 8

We focus on the case where the function H is randomly chosen in a universal hash function family. The case where H is a random oracle is similar to the proof of the Theorem 3.

Let us assume that the scheme is not semantically secure against passive adversaries. Then there is an $\mathsf{IND\text{-}CPA}$ adversary $\mathcal{A}$ that, given the public key pk and the tracing information $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$, can break the scheme with advantage ε. We can then construct an algorithm $\mathcal{B}$ that solves the $\mathsf{DBDH}^2\text{-}\mathsf{E}$ problem. Algorithm $\mathcal{B}$ is given as input a random $\mathsf{DBDH}^2\text{-}\mathsf{E}$ instance $(P, A = aP, B = kP, C = zP, D = ak^2P, U)$ from either the distribution in which U is the $\mathsf{CBDH}^2\text{-}\mathsf{E}$ solution, or the distribution in which U is a random element in $\mathcal{G}_2$. The algorithm $\mathcal{B}$ runs as follows:

Setup: $\mathcal{B}$ sets the public key $\mathsf{pk} = (g, Q = A, Z = g^z = \hat{e}(C, P))$. $\mathcal{B}$ randomly chooses β_0, β_1 and computes:

$$\alpha_0 P = zP - \beta_0 Q \qquad \alpha_1 P = zP - \beta_1 Q.$$

It sends pk, along with $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$ to $\mathcal{A}$.

Challenger: $\mathcal{A}$ outputs two message m_0, m_1 on which it wishes to be challenged. $\mathcal{B}$ picks a random element $b \in \{0, 1\}$ and gives $(A, D, d = m_b \oplus H(U))$ as the challenge to $\mathcal{A}$.

Guess: Algorithm $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. At this point, $\mathcal{B}$ returns 1 if $b = b'$ and 0 otherwise.

Observe that if U is the $\mathsf{CBDH}^2\text{-}\mathsf{E}$ solution, then the challenge ciphertext is an encryption of m_b. Otherwise, since H is randomly chosen from a universal hash function family, the challenge is the ciphertext of a random message, hence $b = b'$ holds with probability $1/2$. By a standard argument, the adversary $\mathcal{B}$ has an advantage of $\varepsilon/2$ in deciding $\mathsf{DBDH}^2\text{-}\mathsf{E}$. $\square$

B.2 Proof of the Theorem 9

From such an adversary $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that breaks the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ problem: Algorithm $\mathcal{B}$ is given as input the $\mathsf{DBDH}^1\text{-}\mathsf{M}$ parameters $(\mathcal{G}_1, \mathcal{G}_2, \hat{e})$ together with a random instance $(P, A = aP, B = kP, X)$ (for $a, k \xleftarrow{R} \mathbb{Z}_q^\star$, $X \xleftarrow{R} \mathcal{G}_1$). Algorithm $\mathcal{B}$ decides whether $X = ak^2P$ by interacting with $\mathcal{A}$

Setup: $\mathcal{B}$ randomly chooses $\alpha_0, \beta_0, \beta_1 \xleftarrow{R} \mathbb{Z}_q^\star$ and computes:

$$zP = \alpha_0 P + \beta_0 A \qquad \alpha_1 P = zP - \beta_1 A \qquad \pi_0 = \beta_0 P \qquad Z = \hat{e}(P, zP).$$

$\mathcal{B}$ sets $Q = A$, $\mathsf{pk} = (g, Q, Z)$ and a proxy (α_0, π_0), as well as the public tracing information $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$. The proxy (α_0, π_0) can be considered as randomly chosen in the set Π_{pk}. Finally, $\mathcal{B}$ gives pk, the proxy (α_0, π_0) and the public tracing information $(\alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$ to $\mathcal{A}$.

Ciphertext: $\mathcal{B}$ randomly chooses $d \in \mathcal{M}$ and builds a ciphertext $(c_1 = B, c_2 = X, d)$. $\mathcal{B}$ sends it to $\mathcal{A}$. Because of the random choice of (B, X, d) and the random choice of the hash function H in a universal hash function family, the challenge (c_1, c_2, d) is a random ciphertext.

Break: If algorithm $\mathcal{A}$ returns $d \oplus H(\hat{e}(\alpha_0 c_1, c_1) \cdot \hat{e}(\pi_0, X))$, $\mathcal{B}$ outputs randomly yes or no. Otherwise $\mathcal{B}$ output no (X is certainly not ak^2P).

Note that when $X = ak^2P$, the ciphertext (kP, X, d) is a random valid ciphertext and the algorithm $\mathcal{A}$ outputs correctly the plaintext $m = d \oplus H(\hat{e}(\alpha_0 c_1, c_1) \cdot \hat{e}(\pi_0, X))$. In this case, the algorithm $\mathcal{B}$ outputs randomly yes or no and the probability that $\mathcal{B}$ gives a correct guess is $1/2$.

When $X \neq ak^2P$, the ciphertext (kP, X, d) is a random invalid ciphertext. Since the decoder behaves differently for invalid ciphertext with probability ε, $\mathcal{A}$ outputs differently than the expected plaintext with probability ε. In such a case, $\mathcal{B}$ answers correctly that X is not equal to ak^2P. In the case $\mathcal{A}$ outputs the expected plaintext (which happens with probability less than $1 - \varepsilon$), $\mathcal{B}$ answers randomly yes or no. Therefore, when $X \neq ak^2P$, the probability that $\mathcal{B}$ gives a correct guess is $\varepsilon + (1 - \varepsilon) \times 1/2 = 1/2 + \varepsilon/2$.

Combining the two above cases, we easily see that $\mathcal{B}$ can solve the DBDH^1-M problem with advantage $\varepsilon/2$. □