This is the Full Version of the Extended Abstract that appears
in the Proceedings of the Second International Conference on Cryptology and Information Security in Latin America (Latincrypt 2012)
(7 – 10 October 2012, Santiago, Chile)
Alejandro Hevia and Gregory Neven Eds., Springer-Verlag, LNCS 7533, pages 56–77.

Message-Based Traitor Tracing with Optimal Ciphertext Rate

Duong Hieu Phan[1,2], David Pointcheval[2], and Mario Strefler[2]

[1] LAGA, University of Paris 8
[2] ENS / CNRS / INRIA

Abstract. Traitor tracing is an important tool to discourage defrauders from illegally broadcasting multimedia content. However, the main techniques consist in tracing the traitors from the pirate decoders they built from the secret keys of dishonest registered users: with either a black-box or a white-box tracing procedure on the pirate decoder, one hopes to trace back one of the traitors who registered in the system. But new techniques for pirates consist either in sending the ephemeral decryption keys to the decoders for real-time decryption, or in making the full content available on the web for later viewing. This way, the pirate does not send any personal information. In order to be able to trace the traitors, one should embed some information, or watermarks, in the multimedia content itself to make it specific to the registered users.
This paper addresses this problem of tracing traitors from the decoded multimedia content or rebroadcasted keys, without increasing too much the bandwidth requirements. More precisely, we construct a message-traceable encryption scheme that has an optimal ciphertext rate, i. e. the ratio of global ciphertext length over message length is arbitrarily close to one.

1 Introduction

Traitor tracing (TT) [CFN94] is a cryptographic primitive used to broadcast content only to a set of authorized users, with an additional tracing property. Unlike broadcast encryption, the set of authorized users is fixed. The two main goals of such a primitive are

- confidentiality: only registered users have access to the broadcast content;
- traceability: if registered users share their secrets to allow unregistered users to access the content, one can trace back at least some of these traitors.

The former property is guaranteed by an encryption procedure, so that only registered users can decrypt and access the content. But an encryption scheme does not prevent users from giving away their secret keys. Even in case several users combine their secret keys in order to make a decryption box (a "pirate decoder"), it should be possible to identify one of the traitors from the code/secrets in the decoder (white-box tracing) or by simply interacting with the decoder (black-box tracing). The *tracing* property should indeed guarantee that even if several users collude to construct a pirate decoder, at least one of the traitors could be found. It should also guarantee non-frameability: an honest user should not be wrongly declared as a traitor.

To circumvent tracing, pirates might try not to make the decoder available, which excludes both white-box and black-box tracing. Instead, they could make only the decrypted content available, or, in case a hybrid encryption scheme is used, the symmetric keys used to encrypt the content: by *message-based traitor tracing*, we aim at tracing traitors from this information only, the decoded content.

Message-based traitor tracing. We propose the term "message-based traitor tracing" as a generic term that subsumes earlier variants and emphasizes the fact that we do not trace from pirate decoders but from information embedded in the content. Fiat and Tassa were the first to consider message-based traitor tracing; in [FT99], they developed *dynamic traitor tracing* to deal with pirates that rebroadcast decrypted content. They assume that there is a real-time feedback from the broadcast content to the center, so that the watermarks can be adapted to the feedback. Safavi-Naini and Wang [SNW00] noted that in this setting, dynamic TT can be prevented by delaying the rebroadcasting of the content. To take this counter-measure into account, they

proposed *sequential traitor tracing*, where the mark allocation is precomputed, but users are removed according to the feedback received. They construct a sequential TT scheme by combining error-correcting codes and watermarking. Jin and Lotspiech [JL07] claimed that protection should not increase the bandwidth by more than 10 %. To solve this problem, they proposed to extend the tracing procedure over several movies (using "inner" and "outer" codes) and assumed that the pirates will not drop any block. Their sequence key block scheme permits the revocation of users after they have been traced through the rebroadcasted messages. Kiayias and Pehlivanoglu [KP09] showed that the sequence key block scheme allows only to trace and to revoke a limited number of users, and proposed a message-trace-and-revoke scheme without this limitation.

Optimal ciphertext rate. Contrary to the classical tracing where schemes with optimal ciphertext rate exist, the problem of constructing a scheme with optimal ciphertext size for message-based traitor tracing is still open. We explain why the solutions for classical tracing fail when applied to message-based traitor tracing and we then describe our approach.

Boneh and Franklin [BF99] developed a traitor tracing scheme with a ciphertext size linear in the maximal number of colluding users. Kiayias and Yung [KY02] further integrated a 2-user version of this scheme with a fingerprinting code into the first TT scheme with a constant ciphertext rate. This method can be summarized as follows. The sender essentially encrypts all the blocks twice, so that the recipient can only decrypt one of the two ciphertexts for each block. The tracing procedure consists in using the decrypted ciphertext or the distributed keys to extract a word associated to the pirate decoder. Thanks to the tracing capability of a fingerprinting code, one can then trace back one of the traitors. Kiayias and Yung's scheme leads to a ciphertext three times bigger than the initial content. Fazio, Nicolosi, and Phan [FNP07] then achieved a ciphertext rate asymptotically 1. Their method is to encrypt just one particular block twice each time and then apply an all-or-nothing transform (AONT), which guarantees that the pirate cannot drop this particular block because missing just one block makes the pirate unable to get any information on the plaintext. The use of AONT in [KY02,FNP07] is interesting but quite impractical because the receiver should wait until he has received n blocks (where n is the code length of the code in use, and thus quite large) to start the decryption procedure. We note that, without aiming to optimize the ciphertext rate, the use of AONT can be avoided by using robust fingerprinting code which allows pirate to drop a fraction of the positions. This is used in [Sir07,BP08,BN08] to reduce the ciphertext size. However, in order to get optimal ciphertext rate in [FNP07], the use of AONT is compulsory, otherwise the pirate could simply drop the particular block to defeat the tracing procedure.

Focusing now on message-based traitor tracing, one natural question is why we do not simply apply the above method of optimizing the ciphertext rate. We argue that this method cannot work for message-based traitor tracing. We first notice that in all the above methods for classical tracing, each user finally gets the same plaintext and if a user redistributes this plaintext, we have no way to trace back the traitor from the distributed message. Therefore, the necessary condition for message-based traitor tracing is that each user receives a different (marked) version of the plaintext. However, when the plaintext is different for each user, one cannot apply AONT for a whole fixed plaintext, otherwise all but at most one user can decrypt. The use of AONT for message-based traitor tracing is thus irrelevant. Fortunately, we can still use the method of doubling one particular block by finding out a way to hide this block. Our method consists in using a 2-user anonymous broadcast encryption scheme and then randomly permuting the blocks. With a 2-user anonymous broadcast encryption scheme, the pirate cannot detect any difference between an encryption for both users (which is used for all blocks but the particular block) and an encryption for one of the two users that is used for the particular protected block. Combining with the permutation of the blocks, we can show that the pirate is prevented from detecting the particular protected block. Moreover, beyond the optimization of ciphertext rate, by not using AONT, our scheme also enjoys the property of the sequential decryption via the use of fingerprinting code as in [BP08,BN08]: the user can sequentially decrypt the sub-ciphertexts, and

does not need to wait to have received the whole ciphertext and to apply the AONT transform to start the decryption procedure.

Our Contribution. Our goal is to improve the technique which consists in distributing two versions of each message block, but without doubling all the blocks. The simplest way, presented in Section 3.1, is to have for each message block m_i two equivalent blocks m_i^0 and m_i^1, so that any sequence $\{m_i^{w_i}\}$, whatever $w \in \{0,1\}^n$ is, corresponds to a valid content m. The two versions m_i^0 and m_i^1 can be provided by either adding watermarks to the original message block m_i, or directly, e. g. by recording a movie with different angles or distances of the shots [BS98]. The blocks, m_i^0 or m_i^1, are both sent over the public channel. However, the user secret keys, usk_i^0 or usk_i^1, have been distributed to the users according to codewords in a fingerprinting code. This means when the authority sees the decoded message m' or the symmetric keys, from each block m'_i, it can tell whether it is m_i^0, m_i^1, or the block has been dropped, and then learns which decryption key has been used: usk_i^0, usk_i^1, or none. From this, it can derive one bit of a word: 0, 1, or 'erasure' respectively. Thanks to the collusion-resistance of the code with erasures, if not too many traitors colluded, at least one of them can be traced back. This method thus consists in encrypting each pair of blocks with two keys. Each user owns only one of the two according to the codeword he received from a fingerprinting code. This results in a ciphertext twice the length of the original message, plus the cost for two key encapsulations per block.

To reduce the length of the encrypted payload, the only way is to protect only a few blocks, not all of them. We present our basic construction in Section 3.2. It resists collusions, but only if the adversary has to output complete messages, i. e. he is not allowed to drop blocks.

If an adversary can detect which blocks are protected because he has access to several different user keys, he can drop some of them without impacting too much the quality of the original message (i. e. a few seconds from a movie). If the adversary can specifically drop protected blocks after decryption, the output contains less information about the keys that were used, which prevents tracing. We thus propose in Section 3.3 an extension of fingerprinting codes to solve this problem so that the adversary cannot specifically erase bits of the codeword: even if we protect 1% of the blocks and the adversary drops 20% of the blocks, he will basically drop only 20% of the protected blocks, and not all of them.

A further improvement, presented in Section 3.4 takes advantage of the fact that some public-key encryption schemes can reuse the randomness in both key encapsulations. For example in ElGamal, given g^r and two public keys $X_0 = g^{x_0}$ and $X_1 = g^{x_1}$, one can derive two sessions keys $Y_0 = X_0^r = g^{rx_0}$ and $Y_1 = X_1^r = g^{rx_1}$. This further reduces the number of group elements needed to encrypt.

This scheme still suffers from long user keys, as we need two key pairs for each message block. In Section 4, we use anonymous broadcast encryption as a primitive instead of PKE to achieve shorter key lengths. We first focus on the two-user case (one message block), which we later extend to cover N users. A message block either consists of a unique message m_i (not protected) or of two versions m_i^0 and m_i^1: in the former case, m_i should be encrypted for the two users, whereas in the latter case, m_i^0 has to be encrypted for user 0, and m_i^1 for user 1. To this aim, we use a 2-user anonymous broadcast encryption scheme (2ABE). Anonymous broadcast encryption allows the selection of any subset of the user set that should be able to decrypt the ciphertext, while hiding who is able to decrypt [LPQ11]. Suppose we have a 2ABE scheme, and we consider ℓ blocks $(m_1, \ldots, m_\ell)$, among which the k-th block only is protected and thus is provided as a pair (m_k^0, m_k^1). We encrypt all the unique blocks m_i for both users, whereas we encrypt m_k^0 for user 0, and m_k^1 for user 1. The ciphertexts are thereafter randomly permuted (but we assume that the message blocks contain indices to reorder them). User 0 and user 1 will both be able to decrypt ℓ ciphertexts among the $\ell + 1$, and after reordering will be able to get the original message. Due to the anonymity, they do not know which block the other user cannot decrypt, therefore they have no idea which block is protected.

The encrypted payload is only $(1 + 1/\ell)$-times as long as the original message, plus the cost of 2ABE key encapsulations, but we need only one key for each user, which is the minimum, given that when viewing the decrypted message, the authority can extract one bit. To achieve full tracing, we extend this case to

an arbitrary number of users by allocating the user secret keys for the 2ABE using our extension of a fingerprinting code [BS98].

2 Definitions

In this section, we define message-based traitor tracing schemes and the building blocks we will use in their construction. We follow an approach similar to [NSS99] by defining first a two-user primitive which we then extend to the multi-user case using fingerprinting codes.

We first state the *marking assumption*, which provides a way to embed a bit in a message block. This will be applied to blocks we protect, whereas no bit will be embedded in non-protected blocks. Then, from the decoded message, the authority will be able to extract the bits involved in the decryption keys in the pirate decoder, unless the decoder drops the protected blocks. We will thus need the property that nobody can detect which blocks are protected so that if the pirate decoder decides to drop some blocks, the choice will be independent from the protection of the blocks. We will show that we can build such a *message-traceable encryption* from a *2-user anonymous broadcast encryption* scheme. Eventually, from all the bits extracted from the protected blocks (and erasures in case of dropped blocks), using the tracing algorithm of a *fingerprinting code*, we can trace back some of the traitors.

2.1 Primitives

As usual, a public-key encryption scheme is defined as a 4-tuple of algorithms $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, along with the standard security notion IND-CPA. A more formal definition of PKE is given in Section A.1 of the appendix. We define an anonymous broadcast encryption scheme as a 3-tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$, along with the standard security notion IND-ACPA. A more formal definition of ABE is given in Section A.2 of the appendix.

2.2 Marking Content

In order to trace from the message content itself, we need to be able to distribute different versions of a message to different users in an undetectable way. One way is to use watermarks. Another way could exploit different camera shots (angle and distance) of the same scene in a movie[BS98]. We abstract away from the concrete way to create versions and use the *marking assumption* that has been introduced by [BS98] to abstract from concrete schemes and has become standard since.[KP10] In the following we assume that, given two blocks m_0 and m_1,

- we can double m_b (for a random $b \in \{0,1\}$) into two *equivalent* messages m_b^0 and m_b^1
- when a user receives m'_0 and m'_1 (such that $m'_b \in \{m_b^0, m_b^1\}$ and $m'_{\bar{b}} = m_{\bar{b}}$, where $\bar{b} = 1 - b$), he cannot guess b.

This essentially means that it is possible to mark a message to protect it, but it is not possible to tell apart protected and unprotected blocks.

In addition, we also assume *robustness* with respect to a symmetric, reflexive relation $\approx_\rho$: for two equivalent blocks $m^0 \approx_\rho m^1$, when the user receives m^b, and tries to alter it (but without changing the meaning or content), he has only a negligible chance to output $m' \approx_\rho m^b$ that is closer to $m^{\bar{b}}$ than to m^b. This reflects that the user cannot change a watermark while preserving the message.

The robustness and the marking assumption guarantee that

- a protected block is indistinguishable from an unprotected block;
- when a user has access to one version of the protected block only, we can learn from its output which bit was embedded: the *detected bit*;

– when a user has access to both versions of the protected block, we either detect from its output one explicit bit as above, or we note that both versions have been used: in either case we can output one bit, associated to at least one version of the block available to the user.

Of course, the user can drop some blocks, but this impacts the quality of the message: we will assume that at most a fraction η of the blocks are dropped.

2.3 Fingerprinting Codes

Fingerprinting codes [BS98] allow an authority to trace a subset of the users (the traitors) that colluded to produce a word (pirate word) from the codewords they were given. This of course depends on the way traitors can derive words from their codewords: the *feasible set* is the set of the useful words that can be derived from the legitimate codewords. We focus on binary codes, defined over the alphabet $\{0,1\}$. In our context, each bit-value is associated to a decryption key, and a receiver has to decrypt at least one block in each pair of variants to be able to get the global content. If all the codewords in a set agree on a position (i. e. they all have the same bit at this position), then the collusion owns only one decryption key, and thus all the words in the feasible set must have the same bit at this position. However, if some of the words differ at a given position, then the collusion owns both decryption keys, and thus both values are possible at this position. More formally, for any list of t words $w_1, \ldots, w_t \in \{0,1\}^n$, $\mathcal{FS}(w_1, \ldots, w_t) = \{w \in \{0,1\}^n \,|\, \forall i \in \{1, \ldots, n\}, \exists j \in \{1, \ldots, t\}, w[i] = w_j[i]\}$.

Definition 1. A *t-fingerprinting code* $\mathcal{T}$ for $\mathcal{FS}$ is defined by a pair of algorithms $(\mathsf{Gen}, \mathsf{Trace})$, where

– $\mathsf{Gen}(N, \varepsilon)$ takes as input the number N of codewords to output and an error probability ε, it outputs a tracing key tk and a code $\Gamma \subset \{0,1\}^n$ of size N.
– $\mathsf{Trace}(\mathsf{tk}, w')$ takes as input the tracing key tk and a word $w' \in \mathcal{FS}(C)$, where C is a collusion of at most t codewords, it outputs a codeword w.

The running time of both algorithms must be polynomial in $N \log(1/\varepsilon)$, and the tracing algorithm should not be wrong too often: with probability less than ε, $w \notin C$.

More precisely, a t-fingerprinting code for $\mathcal{FS}$ guarantees that

– given $(\Gamma, \mathsf{tk}) \leftarrow \mathsf{Gen}(N, \varepsilon)$, with $\Gamma \subset \{0,1\}^n$ of size N
– for any collusion $C \subset \Gamma$ of size at most t, for any $w \in \mathcal{FS}(C)$, $\mathsf{Trace}(\mathsf{tk}, w)$ outputs a word in C with probability $1 - \varepsilon$.

Efficient constructions of such codes can be found in [Tar08]: the resulting code length n for a t-collusion-resistant fingerprinting code is $\mathcal{O}(t^2 \log(N/\varepsilon))$, where ε is the tracing error probability.

2.4 Message-Traceable Encryption

A message-traceable encryption scheme Ψ is a multi-cast encryption scheme which allows all the registered users (with a legitimate secret key) to decrypt a ciphertext. In addition, from the decrypted content, it is possible to derive the key (or even the keys) used for the decryption. In the following description, we focus on static schemes (the maximum number of users is set from the beginning):

– $\mathsf{Setup}(1^\kappa, N, t, \varepsilon)$, where κ is the security parameter, N the number of users, t the maximal size of a collusion, and ε the error probability of the tracing algorithm, generates the global parameters param of the system (omitted in the following), N user secret keys $\{\mathsf{USK}_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N}$, an encryption key EK, and a tracing key TK.

$\mathrm{Exp}^{\mathsf{ind-cca}-b}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon)$
$(\{\mathsf{USK}_{\mathsf{id}}\}, \mathsf{EK}, \mathsf{TK}) \leftarrow \mathsf{Setup}(1^\kappa, N, t, \varepsilon)$; $\mathcal{Q}_D \leftarrow \emptyset$;
$(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot)}(\mathsf{FIND}; \mathsf{EK})$;
$c^* \leftarrow \mathsf{Encrypt}(\mathsf{EK}, m_b)$;
$b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot)}(\mathsf{GUESS}; state, c^*)$;
`if` $c^* \in \mathcal{Q}_D$ `then return` 0 `else return` b';

$\mathsf{ODecrypt}(\mathsf{id}, c)$
$\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{c\}$;
$m \leftarrow \mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, c)$;
`return` m;

Fig. 1. IND-CCA for message-traceable encryption

- $\mathsf{Encrypt}(\mathsf{EK}, m)$ takes as input the encryption key EK and a message m to be sent, it generates a ciphertext c.
- $\mathsf{Decrypt}(\mathsf{USK}, c)$ takes as input a decryption key USK and a ciphertext c, it outputs a message m, or the error symbol $\perp$.
- $\mathsf{Trace}(\mathsf{TK}, c, m)$ takes as input the tracing key TK, a ciphertext c and the decrypted message m, returns an index $\mathsf{id} \in [1, N]$ of a user secret key $\mathsf{USK}_{\mathsf{id}}$.

Security Notions. As for any encryption scheme, the first security notion to define is semantic security, in our case against chosen-ciphertext attacks, whose security game is presented in Figure 1. Of course, to make tracing possible, the encryption algorithm will possibly derive several equivalent versions of the message m_b to be encrypted, which will decrypt to slightly different messages depending on the key used to decrypt. For this reason, we allow the adversary to choose which decryption key should be used by the decryption oracle, hence the additional input id.

Definition 2 (Semantic Security). A message-traceable encryption scheme Ψ is said to be $(\tau, N, t, \varepsilon, q_D, \nu)$-IND-CCA-secure (indistinguishability against chosen-ciphertext attacks) if in the security game presented in Figure 1, the advantage, denoted $\mathrm{Adv}^{\mathsf{ind-cca}}_{\Psi}(\kappa, \tau, N, t, \varepsilon, q_D)$, of any τ-time adversary $\mathcal{A}$ asking for at most q_D decryption queries ($\mathsf{ODecrypt}$ oracle) is bounded by ν.

$$\mathrm{Adv}^{\mathsf{ind-cca}}_{\Psi}(\kappa, \tau, N, t, \varepsilon, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind-cca}-1}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind-cca}-0}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon) = 1]\}$$

This definition includes IND-CPA (for Chosen-Plaintext Attacks) when $q_D = 0$, and thus we denote the advantage $\mathrm{Adv}^{\mathsf{ind-cpa}}_{\Psi}(\kappa, \tau, N, t)$.

We now formalize the additional security notion of *traceability*: after having received at most t secret keys (the collusion C of traitors), the adversary asks for a ciphertext $c^\star$ of a random message $m^\star$, and outputs a plaintext m that should be equivalent to $m^\star$. The tracing algorithm should then output one of the traitors, otherwise the adversary has won the game. We use the relation $m \approx_\rho m'$ from Section 2.2 to denote that two messages are "similar" in the current context. If the adversary sends a random message (hence $m \not\approx_\rho m^\star$) or alternatively outputs an empty message, we say the adversary lost the game:

Definition 3 (Traceability). A message-traceable encryption scheme Ψ is said to be $(\tau, N, t, \varepsilon, \nu)$-traceable if in the security game presented in Figure 2, the success probability, denoted $\mathsf{Succ}^{\mathsf{trace}}_{\Psi,\mathcal{A}}(\kappa, \tau, N, t, \varepsilon)$, of any τ-time adversary asking for at most t secret keys is bounded by ν.

$$\mathsf{Succ}^{\mathsf{trace}}_{\Psi}(\kappa, \tau, N, t, \varepsilon) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{trace}}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon) = 1]\}.$$

3 A Generic Construction from PKE

In this section, we present a series of three simple constructions that illuminate the concepts behind the construction in the next section. The first construction exemplifies a well-known paradigm of constructing

$\mathrm{Exp}^{\mathsf{trace}}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon)$
$(\{\mathsf{USK}_{\mathsf{id}}\}, \mathsf{EK}, \mathsf{TK}) \leftarrow \mathsf{Setup}(1^\kappa, N, t, \varepsilon)$; $\mathcal{Q}_C \leftarrow \emptyset$;
$(state, C) \leftarrow \mathcal{A}(\mathsf{FIND}; \mathsf{EK})$;
$m^\star \xleftarrow{\$} \mathcal{M}$; $c^\star \leftarrow \mathsf{Encrypt}(\mathsf{EK}, m^\star)$;
$m \leftarrow \mathcal{A}(\mathsf{GUESS}; state, c^\star, \{\mathsf{USK}_{\mathsf{id}}\}_{\mathsf{id}\in C})$;
$T \leftarrow \mathsf{Trace}(\mathsf{TK}, c^\star, m)$;
`if` $m = \perp$ `or` $m \not\approx_\rho m^\star$ `then return` 0;
`if` $T \cap C = \emptyset$ `then return` 1 `else return` 0;

Fig. 2. Traceability

a message-based traitor tracing scheme from a PKE scheme and a fingerprinting code, and serves as a stepping stone for the second construction, which shows how to achieve optimal ciphertext rate. The first two constructions are only secure when the adversary is required to retransmit only complete messages. Our third construction modifies the previous one to account for adversaries that drop parts of the message. The fourth construction is an exercise to drive the efficiency of the second construction to its limits by reusing randomness.

3.1 A Simple Construction

To have a baseline against which to compare our later constructions, we first outline a simple way to construct a message-based traitor tracing scheme. The construction uses a PKE scheme Π and assigns user keys according to the codewords of the fingerprinting code $\mathcal{T}$. If the codewords have length n, we need $2n$ instances of the PKE scheme.

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon)$
  1. It generates a t-fingerprinting code, using $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon)$, with a low error value ε. We thus denote n the length of a codeword in Γ, and enumerate codewords with indices associated to each users: $\Gamma = \{w_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^n$.
  2. it then calls $2n$ times $\Pi.\mathsf{Setup}(1^\kappa)$ to obtain $(\mathsf{dk}_i^b, \mathsf{ek}_i^b)_{b=0,1,i=1\ldots n}$.
  3. it sets $\mathsf{EK} \leftarrow \{\mathsf{ek}_i^b\}_{b=0,1,i=1\ldots n}$, $\mathsf{USK}_{\mathsf{id}} \leftarrow (\mathsf{dk}_i^{w_{\mathsf{id}}[i]})_{i=1\ldots n}$ for all $\mathsf{id} \in [1, N]$, $\mathsf{TK} \leftarrow (\{\mathsf{dk}_i^0, \mathsf{dk}_i^1\}_{i=1,\ldots,n}, \mathsf{tk})$.
- $\mathsf{Encrypt}(\mathsf{EK}, m)$ first splits m into n blocks $m_1, \ldots m_n$. For each block,
  1. it creates two versions m_i^0, m_i^1 of each block m_i
  2. it then encrypts the versions as $c_i^b = \Pi.\mathsf{Encrypt}(\mathsf{ek}_i^b, m_i^b)$
  3. it sets $c = (c_1^0, c_1^1, \ldots, c_n^0, c_n^1)$.
- $\mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, c)$
  1. it parses $c = (c_1^0, c_1^1, \ldots, c_n^0, c_n^1)$
  2. it decrypts $m_i^{w_{\mathsf{id}}[i]} = \Pi.\mathsf{Decrypt}(\mathsf{dk}_i^{w_{\mathsf{id}}[i]}, c_i^{w_{\mathsf{id}}[i]})$ to recover m.
- $\mathsf{Trace}(\mathsf{TK}, c, m)$ extracts the word w' from m and calls $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w')$ to get the codeword w_{id} of a colluder.

3.2 Improved Construction

We can reduce the ciphertext rate for long messages by watermarking only some blocks. We now describe a generic construction that accomplishes this by encrypting a message consisting of n sequences of ℓ blocks each in such a way that in sequence i, $\ell - 1$ blocks can be decrypted by both users; these blocks are not used for tracing. The other block is duplicated using to different marks and encrypted at two positions $v_0[i], v_1[i]$, each time for one key only: the message at position $v_0[i]$ cannot be decrypted by users with key 0, and the

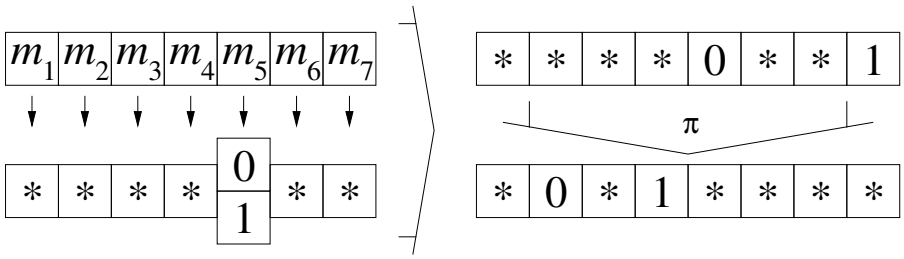

Fig. 3. Hiding a mark at position 5 in a sequence of 7 blocks.

message at position $v_1[i]$ cannot be decrypted by users with key 1. By doing this, the ciphertext will have a length of $(1 + 1/\ell)$-times the length of the message, plus the overhead for encryption.

To reduce the overhead for encryption for long messages, we now model the PKE scheme as a KEM. Given any symmetric cipher $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$, a PKE Π, and a fingerprinting code $\mathcal{T}$, we construct a message-traceable encryption scheme $\hat{\Psi}(\ell, n)$ as follows:

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon, \ell)$:
  1. It generates a t-fingerprinting code, using $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon)$, with a low error value ε. We thus denote n the length of a codeword in Γ, and enumerate codewords with indices associated to each users: $\Gamma = \{w_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^n$.
  2. It then calls $\Pi.\mathsf{Setup}(1^\kappa)$ $n(\ell+1)$ times to obtain, for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell+1$, $\mathsf{ek}_{i,j}, \mathsf{dk}_{i,j}$. It draws two random vectors $v_0, v_1 \in [1, \ell+1]^n$ with the condition that $v_0[i] \neq v_1[i]$ for all $i = 1, \ldots n$. The position $v_b[i]$ describes the secret key that the users with $w_{\mathsf{id}}[i] = b$ do *not* have. We set

$$\begin{aligned}
\mathsf{USK}_{\mathsf{id}} &\leftarrow \{\mathsf{dk}_{i,j}\}_{\substack{i=1,\ldots,n \\ j \neq v_{w_{\mathsf{id}}[i]}[i]}}, \\
\mathsf{EK} &\leftarrow (\{\mathsf{ek}_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell+1}}, v_0, v_1), \\
\mathsf{TK} &\leftarrow (\{\mathsf{dk}^0_{i,j}, \mathsf{dk}^1_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell+1}}, \mathsf{tk}).
\end{aligned}$$

- $\mathsf{Encrypt}(\mathsf{EK}, m)$ first splits m in $n\ell$ blocks $\{m_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell}}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. it chooses a random position $k \in \{1, \ldots, \ell\}$, to protect block $m_{i,k}$;
  2. it generates two equivalent versions $m^0_{i,k}, m^1_{i,k}$, of this block (see Figure 3), resulting in a list of $\ell+1$ blocks;
  3. it prepends the position to the block: $M_j = j \| m_{i,j}$, for $j = 1, \ldots, \ell$, $j \neq k$, $M_k = k \| m^0_{i,k}$, $M_{\ell+1} = k \| m^1_{i,k}$;
  4. it chooses a random permutation $\pi \in S_{\ell+1}$ with the restriction that the position of the marked blocks is $v_1[i] = \pi(k)$ and $v_0[i] = \pi(\ell+1)$; and permutes the blocks: $M'_j = M_{\pi(j)}$.
  5. it generates the session keys: $(c_{i,j}, K_{i,j}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_{i,j})$,
  6. It then encrypts the blocks under the symmetric keys: $C_{i,j} \leftarrow (c_{i,j}, c'_{i,j} = \mathsf{Enc}_{K_{i,j}}(M'_j))$ for $j = 1, \ldots, \ell+1$.

  The final ciphertext consists of all the pairs $C_{i,j}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell+1$.
- $\mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, C)$ takes as input the key $\mathsf{USK}_{\mathsf{id}} = \{\mathsf{dk}_{i,j}\}_{\substack{i=1,\ldots,n \\ j \neq v_{w_{\mathsf{id}}[i]}[i]}}$ and a ciphertext $C = \{C_{i,j}\}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. it calls $\Pi.\mathsf{Decaps}(\mathsf{usk}^{w_{\mathsf{id}}[i]}_{i,j}, c_{i,j})$, for $j \neq v_{w_{\mathsf{id}}[i]}[i]$, to obtain the session key $K_{i,j}$;
  2. it decrypts the message with $\mathsf{Dec}(K_{i,j}, c'_{i,j})$, which outputs $M'_{i,j}$;
  3. it should be able to parse the $M'_{i,j} = p_j \| m_{i,j}$, with $\{p_j\} = \{1, \ldots, \ell\}$, otherwise it stops and outputs $\perp$;

4. it eventually reorders the messages according to p_j, and concatenates the other parts.

It concatenates the ℓ blocks in each sequences, and the n sequence-results to output the full message m.

– $\mathsf{Trace}(\mathsf{TK}, C, m)$ can detect the protected blocks using the decryption keys in TK. From the block that was actually decrypted in each sequence i, it can learn the value of the bit $w[i]$. Then, thanks to the traceability of the code $\mathcal{T}$, the $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w)$ outputs a traitor.

Remark 4. The traceability of the scheme rests on the fact that a user does not know which of the keys are common to all users and which are specific to those with the same bit in the codeword. While a user that shares the information which positions he cannot decrypt with other users is considered to be misbehaving and thus corrupted in our security model, the real-life cost of sharing some of these positions is quite low. The scheme is thus susceptible to a Pirates 2.0-attack as described in [BP09].

3.3 Adapting the Code for Deletions

The above scheme works well if we require that the users output only complete messages. However, in practice removing small parts of a movie might still result in an acceptable quality. In this case, we want to ensure that the adversary cannot drop specifically the blocks that contain watermarks, but that in order to delete a certain fraction of the codeword, he has to delete the same fraction of blocks. If users collude in constructing the message, i. e. the adversary has access to several user secret keys, they can choose to drop only the blocks that contain watermarks. In this case, in the places where the codewords that correspond to the user secret keys differ, the adversary can find the marked message blocks and drop only those. The tracing authority cannot know if a block was dropped at random or because the adversary knew it was marked, so tracing is impossible.

To solve this problem, we design a new code from a fingerprinting code by repeating and permuting the bits of the codeword. If the adversary cannot tell which bits of the new code are repetitions of the same bit, it is unlikely that all repetitions of a bit from the fingerprinting code are erased accidentally. The tracing authority can then assume that for an erased bit, both keys are known to the attacker. The only problem is that colluding users can detect in which sequence they all decrypt the same blocks, which implies that they have the same code bit in this position. They can then drop blocks specifically from sequences where their code bits agree. To prevent this, we insert dummy bits that are the same for all users, which doubles the length of the code.

Let η be the fraction of blocks that the adversary is allowed to drop. We choose an integer ρ such that $\eta^\rho \leq \varepsilon/2n$. Let $w = w[1] \dots w[n]$ be a codeword from the fingerprinting code. We generate a codeword of our new code by repeating each bit of w ρ times, padding it with $n\rho$ dummy bits that are identical for all users, then applying a permutation:
We first describe $\mathsf{Gen}(N, \varepsilon)$:

1. Generate the fingerprinting code: $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon/2)$
2. Choose a random permutation $\pi : \{0,1\}^{2n\rho} \to \{0,1\}^{2n\rho}$
3. $\mathsf{tk}' \stackrel{\mathsf{def}}{=} (\mathsf{tk}, \pi^{-1})$
4. Choose a random string $s \stackrel{\$}{\leftarrow} \{0,1\}^{n\rho}$
5. $\Gamma' \stackrel{\mathsf{def}}{=} \{w'_{\mathsf{id}}\}_{\mathsf{id}=1,\dots,N} \subset \{0,1\}^{2n\rho}$ where $w'_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \pi(w_{\mathsf{id}}[1] \dots w_{\mathsf{id}}[1] \dots w_{\mathsf{id}}[n] \| s)$.
6. Output (Γ', tk').

To trace, run the modified algorithm $\mathsf{Trace}(\mathsf{tk}', w')$:

1. Reconstruct a word w from $\pi^{-1}(w')$:
   (a) If all of the ρ replications of bit $w[i]$ that were not erased are equal to b, set $w[i] = b$.
   (b) If at least one of the replications of bit $w[i]$ has the value 1 and at least one of them has the value 0, choose $w[i]$ at random (in this case the adversary knows both keys).

(c) If all replications of the bit $w[i]$ have been erased, choose $w[i]$ at random.

2. Return the output of $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w)$.

Since a codeword is 2ρ times as long as previously, the user secret keys will also be 2ρ times as long. The tracing key tk contains a secret permutation π^{-1}, so tracing is no longer public as opposed to tracing for the fingerprinting code. However, in our construction we need to include the user secret keys in the tracing key, so in our construction tracing was not public even before the change in the code used.

3.4 Reusing Randomness

To further reduce the ciphertext rate, we can try to reduce the size of the key encapsulation, by reusing the random coins in all the ciphertexts of a sequence or even the complete ciphertext.

We need to make sure that the PKE scheme in the construction remains secure if randomness us reused. If we instantiate the PKE scheme with ElGamal, we can use the results of Bellare, Boldyreva, Kurosawa, and Staddon [BBKS07, Lemma 7.2].

The only change is in the encryption:

– $\mathsf{Encrypt}(\mathsf{EK}, m)$: it first splits m in $n\ell$ blocks $\{m_{i,j}\}_{\substack{i=1,\ldots,n\\ j=1,\ldots,\ell}}$. For each sequence, at position $i \in \{1,\ldots,n\}$:

5. it draws the common randomness $R \xleftarrow{\$} \mathcal{R}$, then generates the session keys: $(c_{i,j}, K_{i,j}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_{i,j}; R)$,

The final ciphertext consists, as above, of all the pairs $C_{i,j} = (c_{i,j}, c'_{i,j})$, but where all the $c_{i,j}$ in the same sequence use a common part that can be included once only.

Reusing randomness incurs a loss in the security reduction for the PKE scheme that is equal to the number of ciphertexts that share the same randomness [BBKS07, Th. 6.2]. This means that in the generic case, the group has to be larger to accommodate the loss in security. In the case of ElGamal however, we can exploit the random self-reducibility to avoid this increase in the group size [BBKS07, Th. 7.3].

3.5 Security

For reasons of space, and since these constructions are not the main result of this paper, we do not give a security proof for them. Given the security proof of our final construction, the security of the above constructions is an easy corollary.

4 A Construction with Shorter Keys

The main disadvantage of the PKE-based construction is the length of the user keys, which must contain a PKE key for each block. Since a codeword has n bits, we can hope to reduce the number of different user keys to n as well. To achieve this, we use a primitive that allows encryption to either of two users or to both of them: 2-user broadcast encryption.[1]

Our message-traceable encryption scheme makes use of codes, where the bits of the codewords are embedded in a message by doubling some parts of it, the so-called *protected blocks*. Because we do not want the adversary to learn which parts of the message contain bits of the codeword, we need a broadcast encryption scheme where a user cannot tell whether a block is destined only for his key or for both keys, a 2-user anonymous broadcast encryption (2ABE).

This requires the symmetric cipher used with this construction to be weakly robust[ABN10], since one of the decapsulated keys will be either $\perp$ or an unusable key. The construction uses one instance of the 2ABE

[1] We view the scheme as an anonymous broadcast encryption scheme because broadcast encryption is a well-known concept that is intuitively understood. In [KY02], the same primitive was called a "2-key, 1-copyrighted public-key encryption scheme".

scheme Π per bit of the codeword, encrypting $\ell+1$ messages at a time in one sequence, with the target sets determined by the positions v, w where the watermarks are embedded. In this construction, the length of the EK and USK is n times that of Π, and to encrypt a sequence of ℓ blocks, doubling one block, we need $\ell+1$ Π key-encapsulations plus $\ell+1$ symmetrically encrypted message blocks.

4.1 Construction of a Message-Traceable Encryption Scheme

Our first construction combines a fingerprinting code $\mathcal{T}$ with a 2ABE scheme Π. If the codewords have length n, we need n instances of the 2ABE scheme. Given any weakly robust[ABN10] symmetric cipher $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$, a 2-user anonymous broadcast encryption Π, and a traceable code $\mathcal{T}$, we construct a message-traceable encryption scheme $\Psi(\ell, n)$ as follows:

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon, \ell)$:
  1. It first generates a t-fingerprinting code, using $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon)$. We thus denote n the length of a codeword in Γ, and enumerate codewords with indices associated to each users: $\Gamma = \{w_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^n$.
  2. It then calls $\Pi.\mathsf{Setup}(1^\kappa, 2)$ n times to obtain, for $i = 1, \ldots, n$, $\mathsf{ek}_i, \mathsf{usk}_i^0, \mathsf{usk}_i^1$. We set

$$\begin{aligned}\mathsf{USK}_{\mathsf{id}} &\leftarrow (\mathsf{usk}_1^{w_{\mathsf{id}}[1]}, \ldots, \mathsf{usk}_n^{w_{\mathsf{id}}[n]}),\\ \mathsf{EK} &\leftarrow (\mathsf{ek}_1, \ldots, \mathsf{ek}_n),\\ \mathsf{TK} &\leftarrow (\{\mathsf{usk}_i^0, \mathsf{usk}_i^1\}_{i=1,\ldots,n}, \mathsf{tk}).\end{aligned}$$

- $\mathsf{Encrypt}(\mathsf{EK}, m)$: it first splits m in $n\ell$ blocks $\{m_{i,j}\}_{\substack{i=1,\ldots,n\\ j=1,\ldots,\ell}}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. it chooses a random position $k \in \{1, \ldots, \ell\}$, to protect block $m_{i,k}$;
  2. it generates two equivalent versions $m_{i,k}^0, m_{i,k}^1$, of this block (see Figure 3), resulting in a list of $\ell+1$ blocks;
  3. it prepends the position to the block: $M_j = j \| m_{i,j}$, for $j = 1, \ldots, \ell$, $j \neq k$, $M_k = k \| m_{i,k}^0$, $M_{\ell+1} = k \| m_{i,k}^1$;
  4. it chooses a random permutation $\pi \in S_{\ell+1}$ and permutes the blocks: $M'_i = M_{\pi(i)}$. We note $v = \pi(k)$ and $w = \pi(\ell+1)$, the positions of the two equivalent blocks;
  5. it generates session keys for all the blocks, except M'_v and M'_w, with the 2ABE scheme Π, with the full target set $\{0,1\}$, whereas M'_v is targeted to $\{0\}$ only, and M'_w is targeted to $\{1\}$ only. More precisely, it first generates the session keys: $(c_{i,j}, K_{i,j}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_i, \{0,1\})$, for $j \neq v, w$, $(c_{i,v}, K_{i,v}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_i, \{0\})$, and $(c_{i,w}, K_{i,w}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_i, \{1\})$.
  6. It then encrypts the blocks under the symmetric keys: $C_{i,j} \leftarrow (c_{i,j}, c'_{i,j} = \mathsf{Enc}_{K_{i,j}}(M'_j))$ for $j = 1, \ldots, \ell+1$.

  The final ciphertext consists of all the pairs $C_{i,j}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell+1$.
- $\mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, C)$ takes as input the key $\mathsf{USK}_{\mathsf{id}} = (\mathsf{usk}_1^{w_{\mathsf{id}}[1]}, \ldots, \mathsf{usk}_n^{w_{\mathsf{id}}[n]})$ and a ciphertext $C = \{C_{i,j}\}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. it calls $\Pi.\mathsf{Decaps}(\mathsf{usk}_i^{w_{\mathsf{id}}[i]}, c_{i,j})$, for $j = 1, \ldots, \ell+1$, to obtain the session key $K_{i,j}$;
  2. it decrypts the message with $\mathsf{Dec}(K_{i,j}, c'_{i,j})$, which outputs either $M'_{i,j}$ or $\perp$ (because of the robustness);
  3. it should be able to parse the $M'_{i,j} = p_j \| m_{i,j}$, with $\{p_j\} = \{1, \ldots, \ell\}$, otherwise it stops and output $\perp$;
  4. it eventually reorders the messages according to p_j, and concatenates the other parts.

  It concatenates the ℓ blocks in each sequences, and the n sequence-results to output the full message m.
- $\mathsf{Trace}(\mathsf{TK}, C, m)$ can detect the protected blocks using the decryption keys in TK. From the block that was actually decrypted in each sequence i, it can learn the value of the bit $w[i]$. Then, thanks to the traceability of the code $\mathcal{T}$, the $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w)$ outputs a traitor.

4.2 Security of the Construction

We show that our construction fulfills IND-CPA-security and explain under which conditions it is traceable.

Theorem 5. *If the 2ABE scheme Π is IND-CPA and the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, then our construction $\Psi(\ell, n)$ is IND-CPA, and*

$$\mathrm{Adv}^{\mathsf{ind-cpa}}_{\Psi(\ell,n)}(\kappa, \tau, N, t, \varepsilon) \leq n \cdot (\ell + 1) \times \left(2 \cdot \mathrm{Adv}^{\mathsf{ind-cpa}}_{\Pi}(\kappa, \tau_1, 2) + \mathrm{Adv}^{\mathsf{ind-cpa}}_{\mathcal{E}}(\kappa, \tau_2)\right).$$

The proof can be found in the Appendix B.1.

Before we turn to tracing, we state an intermediate result. The following lemma says that no adversary can tell which blocks are not encrypted to all users, if he only has one of the two keys, w.l.o.g. usk_i^0.

Lemma 6. *If the 2ABE scheme Π is both IND-CPA and ANO-CPA, and the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, then an adversary who only has the usk_i^0 for a sequence i cannot distinguish between the case where the block at position v is encrypted to the target set $\{0\}$ and the block at position w is encrypted to the target set $\{1\}$ and the case where the block at position v is encrypted to the target set $\{0, 1\}$ and the block at position w contains a random message. If we denote by $\mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa, \tau, N, t)$ the maximal advantage of any adversary within time τ, on any index i, then*

$$\mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa, \tau, N, t, \varepsilon) \leq \mathrm{Adv}^{\mathsf{ano-cpa}}_{\Pi}(\kappa, \tau_1, 2) + \mathrm{Adv}^{\mathsf{ind-cpa}}_{\Pi}(\kappa, \tau_2, 2) + \mathrm{Adv}^{\mathsf{ind-cpa}}_{\mathcal{E}}(\kappa, \tau_3).$$

We prove the lemma in the Appendix B.2.

It follows immediately from the lemma that as long as the message is complete, i. e. no blocks were dropped, any collusion of up to t users can be traced.

We now consider the case where the adversary is allowed to drop blocks. If there are no collusions, the case is unproblematic if robust codes are used (that resist erasures), because if the adversary drops a fraction δ of the blocks, he cannot erase significantly more than a fraction δ of the bits in the codeword.

Theorem 7. *Even if an adversary with a single user secret key can drop a fraction η of the message, if the 2ABE scheme Π is both IND-CPA and ANO-CPA, the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, and the code $\mathcal{T}$ is t-fingerprinting for $\mathcal{FS}^\star$ for a fraction $\delta > \eta$ of erasures, then our construction Ψ is traceable. More precisely, one needs*

$$(\delta - \eta)^2 \geq \frac{1}{2} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa, \tau, N, t, \varepsilon).$$

The proof is in the Appendix B.3. We now show that our improvement from Section 3.3 allows tracing collusions of users even if they can drop blocks.

Theorem 8. *Even if an adversary with a single user secret key can drop a fraction η of the message, if the 2ABE scheme Π is both IND-CPA and ANO-CPA, the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, and the code $\mathcal{T}$ is t-fingerprinting with error $\varepsilon/2$, then our construction Ψ with ρ repetitions of each bit of the codeword and $n\rho$ dummy bits is t-fingerprinting with error at most ε as long as*

$$(2\eta)^\rho \leq \frac{\varepsilon}{2n}.$$

4.3 A 2-user Anonymous Broadcast Encryption Scheme

We now present a concrete instance of a 2ABE scheme to use as a building block in our message-based traitor tracing scheme. We view the 2-key 1-copyrighted public-key encryption scheme of Kiayias and Yung [KY02], as a 2-user 1-collusion-secure anonymous broadcast encryption scheme (2ABE). For ease of exposition, we model the scheme as a KEM.

Let G be a group of prime order q, with a generator g. The public parameters consist of (G, q, g). Since we consider the 2-user case, we drop the N parameter:

Construction in	$\|\mathsf{EK}\|$	$\|\mathsf{USK}\|$	$\|\text{KeyHeader}\|$	$\|\text{CTXT}\|$	Ciphertext Rate
Section 3.1	$2\ell n\mathsf{G}$	$\ell n\mathsf{G}$	$\ell n\mathsf{G}$	$2\ell n\mathsf{B}$	$2+\mathsf{G}/\mathsf{B}$
Section 3.1 + rr.	$2\ell n\mathsf{G}$	$\ell n\mathsf{G}$	G	$2\ell n\mathsf{B}$	$2+1\mathsf{G}/(\ell n\mathsf{B})$
Section 3.2	$2(\ell+1)n\mathsf{G}$	$\ell n\mathsf{G}$	$(\ell+1)n\mathsf{G}$	$(\ell+1)n\mathsf{B}$	$1+1/\ell+(1+1/\ell)\mathsf{G}/\mathsf{B}$
Section 3.3	$4(\ell+1)n\rho\mathsf{G}$	$2\ell n\rho\mathsf{G}$	$2(\ell+1)n\rho\mathsf{G}$	$2(\ell+1)n\rho\mathsf{B}$	$1+1/\ell+(1+1/\ell)\mathsf{G}/\mathsf{B}$
Section 3.4	$2(\ell+1)n\mathsf{G}$	$\ell n\mathsf{G}$	G	$(\ell+1)n\mathsf{B}$	$1+1/\ell+1\mathsf{G}/(\ell n\mathsf{B})$
Section 4	$8n\rho\mathsf{G}$	$4n\rho\mathsf{G}$	$4(\ell+1)n\rho\mathsf{G}$	$2(\ell+1)n\rho\mathsf{B}$	$1+1/\ell+(2+2/\ell)\mathsf{G}/\mathsf{B}$

Table 1. Comparison: G is the bit-length of a group element; B is the bit-length of a message block. |PTXT| is always $\ell n\mathsf{B}$, except for the schemes from Section 3.3 and 4, where it is $2\ell n\rho\mathsf{B}$.

- $\mathsf{Setup}(1^\kappa)$ picks $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q^\times$. For the two user-keys one chooses $d'_0, d'_1 \in \mathbb{Z}_q$, and sets $\mathsf{usk}_u \stackrel{\mathsf{def}}{=} (d_u = \alpha - d'_u \cdot \beta, d'_u)$, for $u = 0, 1$. The encryption key is $\mathsf{ek} \stackrel{\mathsf{def}}{=} \{(f = g^\alpha, h = g^\beta), \mathsf{upk}_0 = h^{d'_0}, \mathsf{upk}_1 = h^{d'_1}\}$.
- $\mathsf{Encaps}(\mathsf{ek}, S; r)$ where $r \in \mathbb{Z}_q^\times$
  - if $S = \{0, 1\}$ then $c = (g^r, h^r), K = f^r$
  - else if $S = \{u\}$ then $r' \xleftarrow{\$} \mathbb{Z}_q^\times$, and $c = (g^r, h^{r'}), K = (f/\mathsf{upk}_u)^r \times \mathsf{upk}_u^{r'}$
- $\mathsf{Decaps}(\mathsf{usk}_u, c)$ computes $K = c_0^{d_u} c_1^{d'_u}$. This is equal to $g^{rd_u} \times h^{r'd'_u} = (f/\mathsf{upk}_u)^r \times \mathsf{upk}_u^{r'}$, which ensures correctness in the second case, where $S = \{u\}$. In the first case, since $r' = r$, both users get the same key f^r.

This is a broadcast encryption, because when $S = \{u\}$, the user $1 - u$ decapsulates differently. Anonymity comes from the fact that a ciphertext is either a Diffie-Hellman pair, when $S = \{0, 1\}$, and a random pair in the other case.

4.4 Security of the 2ABE

Theorem 9. *If solving the DDH problem in the underlying group is hard, then the 2ABE scheme presented in Section 4.3 is ANO-ACPA-secure and*

$$\mathrm{Adv}^{\mathsf{ano-acpa}}_{2ABE}(\kappa, \tau) \leq 4 \cdot \mathrm{Adv}^{\mathsf{ddh}}(\kappa, \tau + \tau').$$

The proof of this theorem is given in the Appendix B.5.

Theorem 10. *If solving the DDH problem in the underlying group is hard, then the 2ABE scheme presented in Section 4.3 is a 2-user IND-CPA-secure BE scheme and*

$$\mathrm{Adv}^{\mathsf{ind-cpa}}_{2ABE}(\kappa, \tau) \leq \mathrm{Adv}^{\mathsf{ddh}}(\kappa, \tau + \tau').$$

The proof of this theorem is given in the Appendix B.6.

5 Conclusion

Table 1 shows a comparison of several ways to do message-traceable encryption. We compare the length of the encryption key ($|\mathsf{EK}|$) and the user secret key ($|\mathsf{USK}|$), and the length of the key header and the symmetric encryption of the plaintext (CTXT), normalizing for a plaintext (PTXT) length (before marking blocks) of ℓn blocks. The ciphertext rate is defined as (|KeyHeader| + |CTXT|) / |PTXT|.

The simplest way is to use any public-key encryption scheme to encrypt each message block twice, described in Section 3.1: two pairs of keys are generated for each message block. Using ElGamal, we have one group element for the key header (the key encapsulation) per message-version block. We can reduce the key

header to one group element by reusing randomness. However, even with randomness reuse (rr), using this method it is impossible to reduce the ciphertext rate below 2 without leaving some part of the message unprotected and exposed to untraceable rebroadcasting, since each message block is encrypted twice with the symmetric keys.

Using our improved construction from Section 3.2, we immediately cut the number of blocks that must be sent almost in half with only a small constant increase in the key header as compared to plain ElGamal. The modifications from Section 3.3 multiply key and ciphertext length by 2ρ, but does not change the ciphertext rate. Reusing the randomness in the ciphertext, described in Section 3.4, again shrinks the key header that must be transmitted with every message.

Our main construction from Section 4 achieves the same asymptotic efficiency as the PKE construction, but with keys that are shorter by a factor of ℓ.

Since these results are all asymptotic, the question arises how our scheme performs in practice. Assume we want to design a scheme for $N = 2^{25}$ users, with a maximum false positive rate of $\varepsilon = 2^{-40}$. We set the collusion threshold to $t = 16$, which means we assume that retrieving the keys from 16 different decoders is prohibitively expensive (16 is the largest collusion threshold considered in [JL07, Fig. 5]). Then the length of the code is $n = d_m t^2 \log(N/\varepsilon)$ for some constant d_m. Blayer and Tassa claim that in most real-world applications, one can find a code with $d_m < 8$ [BT08], giving us a code length of 133 120 bits in exchange for a higher false-negative error rate. We assume that we only want to protect against pirates that rebroadcast at least 97% of the blocks, and set $\eta = 2^{-5}$. Since we need $(2\eta)^\rho \leq \varepsilon/2n$, $\rho = 17$, and we double the code length by padding with dummy bits, then our modified code has a length of 4 526 080 bits. We choose the efficiency parameter $\ell = 20$, so we need to split the film into 90 521 600 blocks.

Assuming a block size of 1 kB, our film needs to be at least 90.5 GB.[2] Because of the doubled blocks, the symmetrically encrypted part would be about 95 GB, and the length of the key header would be about 3.8 GB (assuming 160 bit group elements on a suitable elliptic curve), for a total ciphertext size of 98.8 GB that would comfortably fit on an existing 100 GB BDXL Blu-ray disc. The overhead of 9.2% is already within the acceptable range ([JL07] state 10% as their limit), and as we can expect the size of media formats to grow, the concrete efficiency of our scheme will only increase.

Acknowledgments

This work was supported by the French ANR-09-VERS-016 BEST Project and the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II. The authors would like to thank the anonymous reviewers of the Latincrypt 2012 program committee for their helpful comments.

References

ABN10. Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 480–497. Springer, 2010.

BBKS07. Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Trans. on Info. Theory*, 53(11):3927–3943, November 2007.

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

BN08. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *ACM CCS*, pages 455–470, 2008.

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

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

[2] To relate our ciphertext length to existing storage media sizes, we use SI prefixes.

BS98. Dan Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Trans. on Information Theory*, 44(5):1897–1905, 1998.
BT08. Oded Blayer and Tamir Tassa. Improved versions of tardos' fingerprinting scheme. *Designs, Codes and Cryptography*, 48(1):79–103, July 2008.
CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Y. G. Desmedt, editor, *CRYPTO '94*, volume 839 of *LNCS*, pages 257–270. Springer, 1994.
FNP07. Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *Information Security – ISC 2007*, volume 4779 of *Lecture Notes in Computer Science*, pages 71–88. Springer, 2007.
FT99. Amos Fiat and Tamir Tassa. Dynamic traitor tracing. In Michael Wiener, editor, *CRYPTO' 99*, volume 1666 of *LNCS*. Springer, 1999.
JL07. Hongxia Jin and Jeffery Lotspiech. Renewable traitor tracing: A trace-revoke-trace system for anonymous attack. In *ESORICS*, volume 4734 of *LNCS*, pages 563–577. Springer, 2007.
KP09. Aggelos Kiayias and Serdar Pehlivanoglu. Tracing and revoking pirate rebroadcasts. In *ACNS*, volume 5536 of *LNCS*, pages 253–271. Springer, 2009.
KP10. Aggelos Kiayias and Serdar Pehlivanoglu. *Encryption for Digital Content*, volume 52 of *Advances in Information Security*. Springer, 2010.
KY02. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *Eurocrypt 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, 2002.
LPQ11. Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption. Cryptology ePrint Archive, Report 2011/476, 2011.
NSS99. David Naccache, Adi Shamir, and Julien P. Stern. How to copyright a function? In *PKC*, volume 1560 of *LNCS*, pages 188–196. Springer, 1999.
PPS11. Duong Hieu Phan, David Pointcheval, and Mario Strefler. Security notions for broadcast encryption. In *Applied Cryptography and Network Security 2011*, volume 6715 of *LNCS*, pages 377–394. Springer, 2011. full version available from the author's webpage.
PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. of Cryptology*, 13(3):361–396, 2000.
Sir07. Thomas Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. In J.-P. Tillich D. Augot, N. Sendrier, editor, *Proc. of Workshop on Coding and Cryptography (WCC'07)*, pages 379–388, April 2007.
SNW00. Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 316–332. Springer, 2000.
Tar08. Gábor Tardos. Optimal probabilistic fingerprint codes. *Journal of the ACM*, 55(2), May 2008.

A Definitions

A.1 Public-Key Encryption

Definition 11 (Encryption Scheme). A public-key encryption scheme is a 4-tuple of algorithms $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public (encryption) key ek and the associated private (decryption) key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, m; r)$ produces a ciphertext c on the input message m and the public key ek, using the random coins r (we may omit r when the notation is obvious);
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ decrypts the ciphertext c under the private key dk. It outputs the plaintext, or $\perp$ if the ciphertext is invalid.

The correctness requirement is that we get $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m)) = m$ if $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ for all parameters.

Such an encryption scheme is said to be (t, q_D, ν)-IND-CCA-secure (semantic security against chosen-ciphertext attacks) if in the security game presented in Figure 4, the advantage, denoted $\mathrm{Adv}^{\mathsf{ind-cca}}_{\mathcal{PKE}}(k, t, q_D)$, of any t-time adversary $\mathcal{A}$ asking at most q_D decryption queries to the $\mathsf{ODecrypt}$ oracle is bounded by ν:

$$\mathrm{Adv}^{\mathsf{ind-cca}}_{\mathcal{PKE}}(k, t, q_D) = \max_{\mathcal{A}} \{\Pr[\mathrm{Exp}^{\mathsf{ind-cca}-1}_{\mathcal{PKE},\mathcal{A}}(k) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind-cca}-0}_{\mathcal{PKE},\mathcal{A}}(k) = 1]\}.$$

This definition includes IND-CPA (for Chosen-Plaintext Attacks) when $q_D = 0$.

$\mathrm{Exp}^{\mathsf{ind-cca}-b}_{\mathcal{PKE},\mathcal{A}}(k)$
  $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$;
  $\mathcal{Q}_D \leftarrow \emptyset$, $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
  $(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot)}(\mathsf{FIND}; \mathsf{param}, \mathsf{ek})$;
  $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$;
  $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}}(\mathsf{GUESS}, state; c^*)$;
  `if` $c^* \in \mathcal{Q}_D$ `then return` 0;
  `else return` b';

$\mathsf{ODecrypt}(c)$
  $\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{c\}$;
  $m \leftarrow \mathsf{Decrypt}(\mathsf{dk}, c)$;
  `return` m;

Fig. 4. $\mathcal{PKE}$: Semantic Security against Chosen-Ciphertext Attacks (IND-CCA)

A.2 Anonymous Broadcast Encryption

Anonymous broadcast encryption (ABE) allows to address a message to a subset of the users, without revealing this target set even to users who successfully decrypted the message. We define an ABE as a key encapsulation mechanism (KEM), following the definitions found in [PPS11,LPQ11]:

- $\mathsf{Setup}(1^\kappa, N)$, where k is the security parameter, and N the number of users, generates the global parameters param of the system (omitted in the following), N user secret keys $\{\mathsf{usk}_i\}_{i=1,\ldots,N}$, and an encryption key ek.
- $\mathsf{Encaps}(\mathsf{ek}, S; r)$ takes as input the encryption key ek, the target set $S \subset \{1, \ldots, N\}$, and some random coins r (which are sometimes omitted). It outputs a session key K, and an encapsulation c of K;
- $\mathsf{Decaps}(\mathsf{usk}_i, c)$ takes as input a decryption key and a ciphertext c. It outputs the session key K, or the error symbol $\perp$.

For correctness, we require that for any c that encapsulates a key K for a target set S, if $i \in S$, then $\mathsf{Decaps}(\mathsf{usk}_i, c)$ outputs K. Then, semantic security and anonymity should be satisfied.

Definition 12 (Semantic security). We say that an anonymous broadcast encryption (ABE) scheme Π is (τ, N, q_C, q_D, ν)-IND-ACCA-secure (semantic security against adaptive corruption and chosen-ciphertext attacks) if in the security game presented in Figure 5, the advantage, denoted $\mathrm{Adv}^{\mathsf{ind-acca}}_{\Pi}(\kappa, \tau, N, q_C, q_D)$, of any τ-time adversary $\mathcal{A}$ corrupting at most q_C users ($\mathsf{OCorrupt}$ oracle), and asking for at most q_D decryption queries ($\mathsf{ODecrypt}$ oracle), is bounded by ν:

$$\mathrm{Adv}^{\mathsf{ind-acca}}_{\Pi}(\kappa, \tau, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind-acca}-1}_{\Pi,\mathcal{A}}(\kappa, N) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind-acca}-0}_{\Pi,\mathcal{A}}(\kappa, N) = 1]\}.$$

This definition includes IND-ACPA (for Chosen-Plaintext Attacks) when $q_D = 0$, and thus we denote the advantage $\mathrm{Adv}^{\mathsf{ind-acpa}}_{\Pi}(\kappa, \tau, N, q_C)$. When no corruption is allowed, we denote the advantage $\mathrm{Adv}^{\mathsf{ind-cpa}}_{\Pi}(\kappa, \tau, N)$.

Definition 13 (Anonymity). We say that an anonymous broadcast encryption (ABE) scheme Π is (τ, N, q_C, q_D, ν)-ANO-ACCA-secure (anonymity against adaptive corruption and chosen-ciphertext attacks) if in the security game presented in Figure 5, the advantage, denoted $\mathrm{Adv}^{\mathsf{ano-acca}}_{\Pi}(\kappa, \tau, N, q_C, q_D)$, of any τ-time adversary $\mathcal{A}$ corrupting at most q_C users ($\mathsf{OCorrupt}$ oracle), and asking for at most q_D decryption queries ($\mathsf{ODecrypt}$ oracle), is bounded by ν:

$$\mathrm{Adv}^{\mathsf{ano-acca}}_{\Pi}(\kappa, \tau, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ano-acca}-1}_{\Pi,\mathcal{A}}(\kappa, N) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ano-acca}-0}_{\Pi,\mathcal{A}}(\kappa, N) = 1]\}.$$

This definition includes ANO-ACPA (for Chosen-Plaintext Attacks) when $q_D = 0$, and thus we denote the advantage $\mathrm{Adv}^{\mathsf{ano-acpa}}_{\Pi}(k, \tau, N, q_C)$. When no corruption is allowed, we denote the advantage $\mathrm{Adv}^{\mathsf{ano-cpa}}_{\Pi}(\kappa, \tau, N)$.

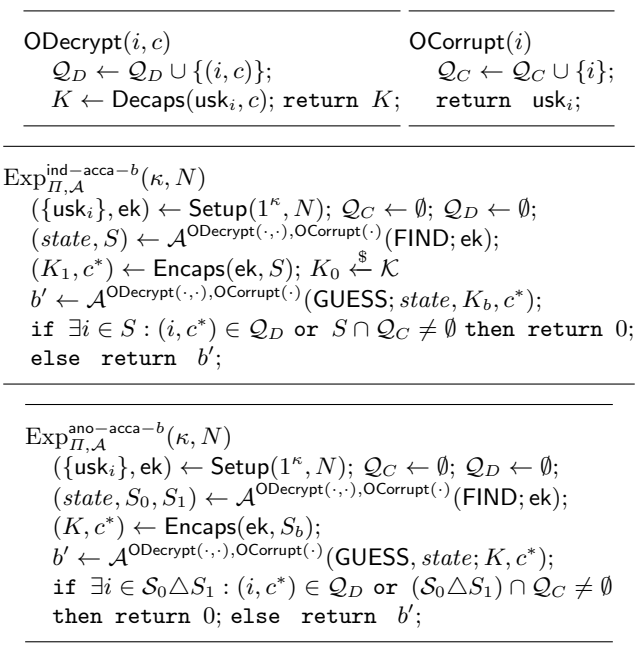

Fig. 5. Security games for ABE

B Proofs

B.1 Proof of Theorem 5

Proof. Let $\mathcal{A}$ be an adversary against the IND-CPA-security of our construction Ψ. We provide a bound on its advantage using a series of games. The simulator first asks for n public keys ek_i, for $i = 1, \ldots, n$, to $\Pi.\mathsf{Setup}(1^\kappa, 2)$, and thus sends the public key EK to the adversary $\mathcal{A}$. The latter sends back two messages $m^0 = (m^0_{1,1}, \ldots, m^0_{1,\ell}, \ldots, m^0_{n,1}, \ldots, m^0_{n,\ell})$ and $m^1 = (m^1_{1,1}, \ldots, m^1_{1,\ell}, \ldots, m^1_{n,1}, \ldots, m^1_{n,\ell})$:

- In game G_0, the simulator encrypts m^0, with k_i the indices of the protected blocks for sequences $i = 1, \ldots, n$, and π_i the permutations;
- In game G_1, the simulator still encrypts m^0, but with random keys for all the key encapsulations of the 2ABE scheme: with an hybrid sequence of games, we can show that the distance between game G_1 and game G_0 is bounded by $n(\ell+1) \cdot \mathrm{Adv}^{\mathsf{ind-cpa}}_{\Pi}(\kappa, \tau_1, 2)$;
- In game G_2, the simulator encrypts m^1, still with random keys for all the key encapsulations of the 2ABE scheme: with an hybrid sequence of games, we can show that the distance between game G_2 and game G_1 is bounded by $n(\ell+1) \cdot \mathrm{Adv}^{\mathsf{ind-cpa}}_{\mathcal{E}}(\kappa, \tau_2)$;
- In game G_3, the simulator encrypts m^1, with k_i the indices of the protected blocks for sequences $i = 1, \ldots, n$, and π_i the permutations: with an hybrid sequence of games, we can show that the distance between game G_3 and game G_2 is bounded by $n(\ell+1) \cdot \mathrm{Adv}^{\mathsf{ind-cpa}}_{\Pi}(\kappa, \tau_1, 2)$;

This concludes the proof.

B.2 Proof of Lemma 6

Proof. Let $\mathcal{A}$ be an adversary who can distinguish the two cases for our construction Ψ. We provide a bound on its advantage using a series of games. The simulator first asks for n public keys ek_i, for $i = 1, \ldots, n$, to $\Pi.\mathsf{Setup}(1^\kappa, 2)$, and thus sends the public key EK to the adversary $\mathcal{A}$, together with $\mathsf{USK}_{\mathsf{id}}$, that corresponds

to a codeword w_{id}. The latter sends back a message $M = (M_{1,1}, \ldots, M_{1,\ell}, \ldots, M_{n,1}, \ldots, M_{n,\ell})$. For all the sequences, except the i-th sequence, for any i, the simulator does as in the real encryption of message M, but we denote m the sequence $M_{i,1} \ldots M_{i,\ell}$.

- In game G_0, the simulator encrypts m, with k the index of the protected block, and (v, w) the positions of the specific ciphertexts to key 0 and key 1 respectively.
- In game G_1, the simulator encrypts m, with k the index of the protected block, and (v, w) the two above positions, but v-th block is encrypted for the two keys, and w-th block is encrypted for the key 1 only. Since the simulator can choose the index v to be the same as the one asked to the anonymity-game, in an indistinguishable way, the distance between game G_1 and game G_0 is bounded by $\mathrm{Adv}_{\Pi}^{\mathsf{ano-cpa}}(\kappa, \tau_1, 2)$;
- In game G_2, the simulator encrypts m, with k the index of the protected block, and (v, w) the two above position, but v-th block is encrypted for the two keys, and a random session key is used for the w-th block, while a key encapsulation is sent for key 1 only. Under the semantic security of the 2ABE scheme, since the adversary does not know key 1, the two games are indistinguishable: the distance between game G_2 and game G_1 is bounded by $\mathrm{Adv}_{\Pi}^{\mathsf{ind-cpa}}(\kappa, \tau_2, 2)$;
- In game G_3, the simulator encrypts m, with k the index of the protected block, and (v, w) the two above position, but the v-th block is encrypted for the two keys, and at position w, a truly random block (with no position appended) is encrypted under a random session key, while a key encapsulation is sent for key 1 only. Under the semantic security of $\mathcal{E}$, the two games are indistinguishable: the distance between game G_3 and game G_2 is bounded by $\mathrm{Adv}_{\mathcal{E}}^{\mathsf{ind-cpa}}(\kappa, \tau_3)$.

B.3 Proof of Theorem 7

Proof. We first fix some notation. Let $\eta_{i,j} \stackrel{\mathsf{def}}{=} \Pr[\mathit{drop}(i, j)]$ be the probability that block j in sequence i is dropped. We denote by $\eta \stackrel{\mathsf{def}}{=} E[\eta_{i,j}]$ the global average probability of a block to be dropped. This means that overall, $n\ell\eta = \sum_{i,j} \eta_{i,j}$ blocks will be dropped. We also consider the average in a sequence i: $\eta_i \stackrel{\mathsf{def}}{=} E[\eta_{i,j}]$. Then $\eta = E[\eta_i]$. Now we consider only the protected blocks. Analogously, let $\theta_i \stackrel{\mathsf{def}}{=} \Pr[\mathit{drop}(i, k_i)]$ be the probability for the protected block in sequence i to be dropped. The average probability of a protected block to be dropped is $\theta \stackrel{\mathsf{def}}{=} E[\theta_i]$. This means that the global number of erasures (dropped protected blocks) is $n\theta = \sum_i \theta_i$. We also consider the gap between protected blocks and any block, by first defining $\gamma_i \stackrel{\mathsf{def}}{=} \theta_i - \eta_i$. We additionally define $\gamma \stackrel{\mathsf{def}}{=} E[\gamma_i] = \theta - \eta$, which we want to show to be small.

Let us define the subset of the sequences in which the gap γ_i is greater than $\gamma - \alpha$ for some α: $I \stackrel{\mathsf{def}}{=} \{i | \gamma_i \geq \gamma - \alpha\}$. We choose a random sequence index $i \in \{1, \ldots, n\}$, and by the splitting lemma [PS00, lemma 1], we know that $\Pr[i \in I] = \Pr[\gamma_i \geq \gamma - \alpha] \geq \alpha$, so with probability greater than α, the gap $\gamma_i = \theta_i - \eta_i$ between the probabilities for the protected block and any block in sequence i to be dropped is greater than $\gamma - \alpha$.

We now focus on this sequence i and the sub-message $m = m_1 \ldots m_\ell$, where m_k is the protected block. We consider the probability of the adversary to drop block k. If the adversary drops the block k, the simulator outputs 1, otherwise the simulator outputs 0. When interacting with the real scheme, the advantage of the simulator (defined as $\Pr[1 \leftarrow \mathcal{S}] - \Pr[0 \leftarrow \mathcal{S}]$) in this game is $\theta_i - (1 - \theta_i) = 2\theta_i - 1$. If we change the scheme so that now m_k is encrypted to both parties, and instead of doubling it, a random block is added, then the advantage is $2\eta_i - 1$.

By lemma 6, the difference between these games is bounded by $\mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t)$. We conditioned on $i \in I$, or, equivalently, $\theta_i - \eta_i = \gamma_i \geq \gamma - \alpha$, which happens with probability α.

$$\begin{aligned} 2\theta_i - 1 - (2\eta_i - 1) &\leq \frac{1}{\alpha} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) \\ 2(\gamma - \alpha) &\leq \frac{1}{\alpha} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) \\ \gamma &\leq \alpha + \frac{1}{2\alpha} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) \\ \theta &\leq (\alpha + \eta) + \frac{1}{2\alpha} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t). \end{aligned}$$

Then, except when it drops blocks, the adversary outputs messages for the protected blocks, and under the assumptions we did on the marking of contents, we can detect a bit. Furthermore, the detected bit follows the rule for the feasible set $\mathcal{FS}$, and erasures (dropped blocks) extend it to $\mathcal{FS}^{\star}$. Since the code $\mathcal{T}$ is (N,t,ε,n)-traceable for $\mathcal{FS}^{\star}$ for a fraction δ of erasures, then our construction Ψ is traceable with tracing-error probability less than ε if the average fraction θ of dropped protected blocks is less than δ: this is, if it exists α such that

$$\begin{aligned} (\alpha + \eta) + \frac{1}{2\alpha} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) &\leq \delta \\ \alpha + (\eta - \delta) + \frac{1}{2\alpha} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) &\leq 0 \\ 2\alpha^2 - 2(\delta - \eta)\alpha + \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) &\leq 0. \end{aligned}$$

To find the minimum, we take the derivative:

$$\begin{aligned} 4\alpha - 2(\delta - \eta) &= 0 \\ \alpha &= \frac{\delta - \eta}{2}. \end{aligned}$$

This is possible as soon as

$$\begin{aligned} 2\alpha^2 - 2(\delta - \eta)\alpha + \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) &\leq 0 \\ \frac{(\delta - \eta)^2}{2} - (\delta - \eta)^2 + \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) &\leq 0 \\ \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t) &\leq \frac{(\delta - \eta)^2}{2}. \end{aligned}$$

This means that we can trace as long as we choose

$$\delta \geq \eta + \sqrt{2\mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t)}.$$

B.4 Proof of Theorem 8

Proof. We need to show that the tracing algorithm works. We accomplish this by showing that from any word w' we can reconstruct a word w that we can use with $\mathcal{T}.\mathsf{Trace}$ to identify a traitor. We inspect the reconstruction of bit $w[i]$ from its ρ repetitions $w[i]_1, \ldots w[i]_\rho$ that can take values in $\{0,1,\star\}$ and define two events where the tracing fails. Let $C = \{w_1, \ldots w_t\}$ denote the set of codewords that correspond to the corrupted users. We define $\mathcal{FS}(C[i])$ to be the set $\{w_1[i], \ldots w_t[i]\}$ of the i-th bits, which can be either of $\{0\}, \{1\}, \{0,1\}$.

- Event E_a is the case if there is an index $j \in [1, \rho]$ with $w[i]_j \neq \star$ and $w[i]_j \notin \mathcal{FS}(C[i])$.
- Event E_b is the case if all $w[i]_j$ are equal to $\star$ and $\mathcal{FS}(C[i]) \neq \{0, 1\}$

Event E_a only happens if the adversary inverted the encryption or broke the watermarking assumption. Event E_b with probablity at most $\varepsilon/2n$. Due to lemma 6 and the random choice of π, the best strategy of the adversary is to drop any block in a sequence where he does not have both keys with probability η. Since half the code bits are dummy bits, he can at most double the fraction of dropped blocks from η to 2η by concentrating the deletions in this way. Therefore, the probability that all ρ copies of $w[i]$ are dropped is $(2\eta)^\rho \leq \varepsilon/2n$.

It follows that the error probability for each of the n bits of the codeword w is at most $\varepsilon/2n$, so the error probability of the tracing algorithm is at most $n(2\eta)^\rho + \varepsilon/2 \leq \varepsilon$.

B.5 Proof of Theorem 9

Proof. The simulator is given a tuple $(g, A = g^a, B = g^b, C = g^c)$. First the simulator will guess the target set. There are only three possibilities (up to the order of the target sets). The simulator chooses

1. $\{0\}, \{1\}$ with probability 1/2.
   In this case, we implicitly set $d'_u = a$ for a user u, and $r = b$ for the challenge random coins. The simulator proceeds as follows. It first flips a bit u to determine where it will embed the challenge. It chooses random $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q^\times$, and defines $f = g^\alpha, h = g^\beta$. It generates $\mathsf{usk}_{1-u} = (d_{1-u}, d'_{1-u})$ as usual, and thus $\mathsf{upk}_{1-u} = h^{d'_{1-u}}$, but $\mathsf{upk}_u = A$. It thus defines $\mathsf{ek} = ((f, h), \mathsf{upk}_0, \mathsf{upk}_1)$. The simulator then draws a random r' and sets the challenge ciphertext to $(B, h^{r'})$. It also returns $K = B^\alpha A^{r'}/C$.
   If the tuple (g, A, B, C) was a DH tuple, we have a ciphertext $c = (g^b, h^{r'})$ and a key $K = g^{b\alpha + ar' - ab} = (f/\mathsf{upk}_u)^b \times \mathsf{upk}_u^{r'}$, that is for $S = \{u\}$. If the tuple was a random tuple, then the key is random.
2. $\{0\}, \{0, 1\}$ with probability 1/4.
   In this case, we implicitly set $\beta = a$, and $r = b$ for the challenge random coins. In this case, the simulator knows that only user 0 can be corrupted, so it chooses the secret key for user 0 in advance and uses it to compute a group element that implicitly has a matching α.
   The simulator proceeds as follows. It sets $h = A$, chooses random $d_0, d'_0 \xleftarrow{\$} \mathbb{Z}_q$ and computes $f \stackrel{\mathsf{def}}{=} g^{d_0} A^{d'_0}$. It sets $\mathsf{usk}_0 = (d_0, d'_0)$, $\mathsf{upk}_0 = A^{d'_0}$, and $\mathsf{upk}_1 = X$ for a random group element X. It thus defines $\mathsf{ek} = ((f, A), \mathsf{upk}_0, \mathsf{upk}_1)$. The challenge ciphertext is set to $c = (B, C)$, for the key $K = B^{d_0} C^{d'_0}$.
   If the tuple (g, A, B, C) was a DH tuple, we have a ciphertext $c = (g^b, g^{ab} = h^b)$ and a key $K = g^{bd_0 + abd'_0} = g^{b(d_0 + ad'_0)} = f^b$, that is for $S = \{0, 1\}$. If the tuple was a random tuple, then $c = (g^b, g^{ab'} = h^{b'})$ where $b' = c/a \neq b$ and a key $K = g^{bd_0 + ab'd'_0} = g^{b(d_0 + ad'_0)} \times g^{ad'_0(b'-b)} = (f/\mathsf{upk}_0)^b \times \mathsf{upk}_0^{b'}$, that is for $S = \{0\}$.
3. $\{1\}, \{0, 1\}$ with probability 1/4.
   This case is done analogously to case 2, exchanging user 0 and 1.

B.6 Proof of Theorem 10

Proof. The simulator is given a tuple $(g, A = g^a, B = g^b, C = g^c)$.

If there is no corruption, a ciphertext is $c = (g^r, h^r)$ for a key f^r. We can implicitly set $h = g^x$ for a known x, $r = a$, $f = B$, $c = (A, A^x)$ and $K = C$, which is a real key if C is the actual Diffie-Hellman value, or a random key otherwise.

In case of corruption, we proceed as in the cases 2 and 3 in the proof of theorem 9, where u is the user for which we know the secret key for corruption.