Trials of Traced Traitors

(Extended Abstract)

Birgit Pfitzmann*

Universität Hildesheim, Institut für Informatik
D-31141 Hildesheim, Germany
pfitzb@informatik.uni-hildesheim.de

Abstract. Traitor tracing schemes as introduced by Chor, Fiat, and Naor at Crypto '94 are intended for tracing people who abuse a broadcast encryption scheme by allowing additional, illegitimate users to decrypt the data. The schemes should also provide legal evidence for such treachery.

We discuss and improve the quality of such evidence, i.e., the security of trials that would be held about supposedly traced traitors. In particular, previous traitor tracing schemes are symmetric in the sense that legitimate users of the broadcast information share all their secrets with the information provider. Thus they cannot offer non-repudiation. We define asymmetric traitor tracing schemes, where the provider, confronted with treachery, obtains information that he could not have produced on his own, and that is therefore much better evidence. Examples of concrete constructions are given.

We also discuss the general model of traitor tracing and propose improvements to the symmetric schemes.

1 Introduction

Traitor tracing schemes were introduced in [ChFN_94]. Their goal is copyright protection for broadcast information, based on broadcast encryption.

1.1 Broadcast Encryption

Broadcast encryption is intended for applications where an *information provider* broadcasts a lot of information in encrypted form, and only several legitimate *users* are supposed to be able to decrypt it. The typical example is Pay-TV, if the information is actually transmitted over a broadcast channel. Broadcast encryption schemes have at least been considered in [Blom_80, Berk_91, FiNa_94, BlCr_95].

The broadcast encryption schemes in [ChFN_94] have three phases:

- *Provider initialization*, where the information provider generates some information that he will need with all users. We call it the *initial record*.
- *User initialization*, where an individual user registers with this information provider. The information that the user stores after this phase is called this user's *personal key*. The information provider may update his initial record after each user initialization.
- *Session sending*. Now data are transmitted, divided into smaller parts called sessions. Each session is encrypted with a different *session key*, and some additional information

* Future address Universität Dortmund, Informatik VI, D-44221 Dortmund 50, Germany

is broadcast that allows all the legitimate users to decrypt the session key with their personal keys, and thus to decrypt the real data.

Other broadcast encryption schemes have an additional *purchase phase* between user initialization and session sending, where one or more users decide that they actually want to decrypt some information, e.g., to watch the coming movie, and the information provider enables precisely these users to decrypt the coming information [FiNa_94]. In contrast, in [ChFN_94] all initialized users are legitimate users of all the information.

Note that the model of broadcast encryption does not allow interaction between users, neither in initialization nor later. This is realistic for applications like Pay-TV, and it is one of the features that distinguish broadcast encryption from conference key distribution (see, e.g., [BDHK_93]).

1.2 Tracing Traitors

The additional goal in [ChFN_94] that extends broadcast encryption to traitor tracing is that legitimate users should not be able to make the data accessible to outsiders without being traced and finally convicted. Users who redistribute data are called *traitors*, outsiders who use this information are called *pirates*, and the pirates' devices are called *pirate decoders*.

The goal of a traitor tracing scheme for broadcast encryption is quite similar to what is often called fingerprinting with other types of data (see, e.g., [Wagn_83, BlMP_86, Caro_95, BoSh_95, ZhKo_95] and these proceedings), except that it is not the broadcast data themselves that are fingerprinted, but, at least in principle, a virtual cryptographic key that is used to encrypt the broadcast data. Before explaining this, however, we discuss the goals of traitor tracing more precisely from the point of view of the application.

Cleartext Redistribution Not Addressed — What Exactly Does This Mean? Traitor tracing schemes do not address treachery where the traitors retransmit the broadband cleartext they obtain to outsiders. This is in contrast to normal fingerprinting, which does address the situation where traitors redistribute the pictures, texts, or programs they obtain. The explanation for this restriction is that operating a pirate TV-broadcast station is deemed too expensive and too risky. More precisely, the information provider will have to perform risk management in his choice between fingerprinting of the actual data and traitor tracing: Fingerprinting is more secure, but does not allow the use of a cheaper broadcast distribution medium. Broadcast encryption with traitor tracing would therefore typically be chosen if the value of the information itself is not very high compared with the price of its distribution.

The formalization of the restriction that traitor tracing does not address cleartext redistribution in Definitions 1 and 2 in [ChFN_94] is that the traitors only redistribute information obtained during user initialization. This is a much stronger restriction; it permits that traitors cannot be traced if they redistribute a tiny amount of data obtained after user initialization. Such a definition may fit applications where timeliness is crucial and the traitors may not be able to redistribute even a tiny amount of data fast enough. For other applications, where pirates can buffer the received encrypted data until the traitors can send them the necessary information, the definition should be extended to include workfactors as parameters. A particularly important one is the ratio between the additional bandwidth the information provider needs on the broadcast channel and the bandwidth the traitors need to remain untraceable. Such parameters can immediately be used in the information provider's risk management.

With the concrete schemes proposed so far (see Section 2.1), the traitors have to redistribute one session key per session to remain untraceable. For the example session length of one minute and a normal encryption system, they would certainly not need a pirate TV-broadcast station, but of course quite some organization. More generally, the ratio between the additional bandwidth the information provider needs on the broadcast channel and the bandwidth the traitors need to distribute the session keys is still quite large and may prevent the use of the schemes, e.g., for encrypted CD ROMs.

No Tamper Resistance Needed. It is not assumed in traitor tracing that the personal keys are in tamper-resistant boxes, which would prevent the traitors from redistribution a priori. One of the reasons is that good tamper-resistant boxes are deemed too expensive. Thus it is assumed that the traitors have found access to their own personal keys and can redistribute them. The goal is to *trace* them if they do so, both for punishment after the fact and for deterring them from redistribution in the first place.

What Input is Needed for Tracing? Of course, tracing is impossible if the redistribution remains entirely secret. In other words, it has to be assumed for tracing that the information provider has found some redistributed information or a pirate decoder. This is quite a reasonable assumption for large-scale redistribution, which is most feared by information providers.

In most cases in practice, the information found will be a decryption key that has some close relation to the personal keys it was derived from. It is therefore both realistic and helpful for understanding to start with such an assumption when describing a tracing algorithm. However, it is also worthwhile considering the more general case, because such an assumption means three things: that the pirate decoder is not too tamper-resistant, that its contents are not too scrambled, and that the encryption scheme does not allow traitors to construct pirate decoders that contain information of a different form and still decrypt successfully.

It was already mentioned in [ChFN_94] that the information provider only needs to use the pirate decoder as a black box, i.e., without opening it, but details were omitted. (The actual procedure sketched below is the same as the authors had in mind.) Thus we mention a few such details, in particular because the question becomes more interesting in the asymmetric case, where the information provider knows less about the users' keys, and thus the information he obtains by experimenting with a black-box pirate decoder will not always be keys.

We assume that the pirate decoder analyzed as a black box does not defy analysis by access control, e.g., by requiring a password before operating. Having found a pirate decoder, but not being able to operate it at all, counts as not having found it. In reality, this means that a pirate cooperates with the information provider against a traitor, willingly or not.

Nevertheless, if the pirate decoder is given in hardware, we do not yet see a procedure that works if the pirate decoder shuts down if it is given inputs that it can identify as incorrectly constructed. As a countermeasure, the information provider might intersperse incorrect data into the normal broadcast data to shut down such pirate decoders while the pirates use them. However, this measure has to be applied with caution, because from the point of view of the honest users this could also be the result of an attack on the broadcast channel.

If the pirate decoder is given in software, the information provider has the additional possibility to reset it to its initial state, and this will also be possible with many hardware pirate decoders. This is the model used below. Furthermore, we will assume for simplicity that the pirate decoder always outputs the correct cleartext if it gets correct data, i.e., it does not take measures that might fool an analyzer, but would also disturb the pirate.

Another Approach. An alternative to traitor tracing is presented in [DwLN_96], so-called signets. In these schemes, the information provider does not trace a traitor at all; he need not even find a pirate decoder. Instead, a traitor has to either divulge some private information to the pirates or distribute a fairly large amount of data. However, the same private information must be divulged to the information provider, which makes the model underlying this cryptographically very interesting scheme somewhat doubtful. (The main proposal is to use credit card numbers. This does indeed fit the widespread understanding of credit card numbers as secrets, although they are divulged to hundreds of merchants and on the telephone, but I would prefer to have this understanding changed than utilized further.) One can, however, try to combine those techniques with the following asymmetric ideas, so that the private information can also remain private from the information provider.

1.3 Asymmetric Traitor Tracing

The main purpose of this paper is to introduce *asymmetric traitor tracing*. This is similar to asymmetric fingerprinting, recently introduced in [PfSc_96]. The basic observation is that previous traitor tracing schemes are symmetric in the sense that legitimate users share all their secrets with the information provider. Thus when seemingly redistributed information is found, it could just as well have been produced by the information provider himself, or by dishonest personnel or someone with illegal access to the information provider's equipment, as by a traitor. The result of tracing is therefore no real evidence that could unambiguously convince a third party, just as symmetric message authentication codes do not provide non-repudiation, in contrast to asymmetric digital signature schemes.

The asymmetric traitor tracing schemes introduced here avoid this problem: The information provider, confronted with treachery, obtains information that he could not have produced on his own, and that is therefore real evidence of the treachery.

1.4 Fully Frameproof Symmetric Schemes

Our main contribution to symmetric traitor tracing is to introduce and construct fully frameproof schemes.

In [ChFN_94], there is only one parameter k for collusion sizes, i.e., the maximum number of traitors against which the schemes are secure. More than k traitors can not only redistribute information without being traced, but also *frame* an honest user, i.e., make this user seem a traitor. This seems quite unacceptable, at least if third parties sometimes do believe an accusation by the information provider even though the scheme is only symmetric. At least, framing is far worse than undetected redistribution, which underlies the normal risk management by the information provider. Moreover, the maximum collusion size is quite small in practice because of the performance of the existing constructions; $k = 32$ is mentioned as a realistic example in [ChFN_94]. Thus the traitors might, e.g., be a school class. The definition would even allow them to selectively frame their teacher. (The constructions do not.) One may argue that the class could also claim to have watched their teacher shoplifting, but in that case it is at least clear who is making the accusation, and they can be questioned, whereas in the traitor tracing schemes, they remain anonymous.

Thus we propose that there should be a separate, much larger parameter k_{frame} for the size of collusions that cannot frame. We call a scheme *fully frameproof* if no collusion of traitors can frame another user, i.e., if $k_{frame} = N$, where N is the number of users. Of course, even fully frameproof symmetric schemes do not prevent the information provider from framing a user. They can be seen as an intermediate step between previous symmetric schemes, where small collusions of users can frame someone, and asymmetric schemes, where even arbitrary collusions including the information provider cannot frame anyone.

2 The Chor-Fiat-Naor Schemes

We first sketch the constructions from [ChFN_94], which are the basis of all our constructions.

2.1 Use of Session Keys

Two items are central to all these schemes:

- Each session is encrypted with an ordinary symmetric encryption scheme under a new, randomly generated *session key*. We assume that a symmetric encryption scheme has been fixed.
- Before each session, a so-called *enabling block* is sent. This may happen on a separate signaling channel so that the broadcast transmission is not interrupted. The enabling block contains an encryption of the next session key. This encryption is done in a complicated way that allows traitors to be traced.

Before going into details of this encryption, we can already substantiate two claims made in Section 1.2.

First, we can see that traitors who redistribute session keys instead of their personal keys cannot be traced, because the session key is the same for all.

Secondly, we can see how such traitor tracing schemes can be partitioned into fingerprinting of a key and normal encryption. Fingerprinting normal information, in particular texts and images, means that each user obtains a slightly different copy, where "slightly different" usually primarily means that the difference is not perceptible by humans. The intrinsic meaning is some semantic equivalence or at least similarity of copy and original. With decryption keys, the natural semantic equivalence is the ability to decrypt the same information. Thus we can see the following as fingerprinting a virtual master key that decrypts all the session keys, where the decryption function is also modified by fingerprinting.

2.2 Overview of Key Fingerprinting

We restrict ourselves to the so-called one-level schemes from [ChFN_94]. They have the following common elements, illustrated in Figure 1:

- During provider initialization, the information provider generates l sets of b keys each, where l and b are parameters to be determined later for each individual scheme. (Two necessary conditions are $l > k$ and $b^l > N$; recall that k is the maximum number of traitors and N the number of users.) These keys also belong to the given symmetric encryption scheme. We call each such set a *bucket*. The keys in the i-th bucket are denoted $key_{i,1}, \ldots, key_{i,b}$.
- During user initialization, each user gets one key from each bucket. (The trick is which ones.) We call the indices of these keys the user's *codeword*, i.e., a codeword $(c_1, \ldots, c_l) \in \{1, \ldots, b\}^l$ means that this user has the keys $key_{1,c_1}, \ldots, key_{l,c_l}$.
- Each session key s is the exor (the bitwise exclusive or), of l random values $s_1, \ldots, s_l$. We call them the *shares* of the session key. For all i, the enabling block contains the encryptions of s_i under each key from the i-th bucket. As each user has one key from the i-th bucket, she can decrypt one of these b encryptions of the i-th share, and finally reconstruct the session key.

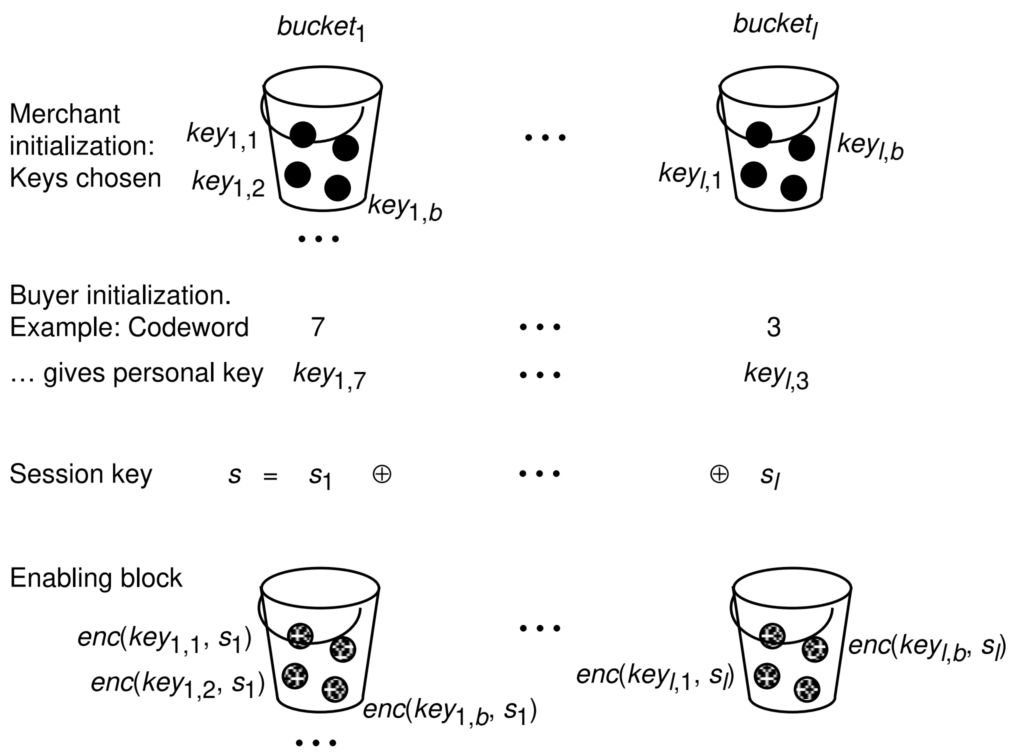

Figure 1 One-level traitor tracing scheme

- The simplified assumption for tracing is that the information provider has opened a pirate decoder and found one key from each bucket. He may find more, of course; then he selects, say, the lexicographically first one.

  If the pirate decoder is only given as a black box with resetting, a procedure of the following type is used to identify the index of one key in bucket i: For $j = 1, \ldots, b - 1$, the information provider makes many trials where the first j shares for this bucket are random and the others correct. For each trial, the decoder is reset to its initial state. If the observed probability of correct decryption for j is smaller than that for $j - 1$ by a certain threshold, the information provider concludes that $key_{i,j}$ is present.

- The idea for tracing is as follows: If up to k traitors had to provide l keys for a pirate decoder, at least one of them must have provided at least l/k keys. The information provider simply looks from which user the largest number of keys is present in the pirate decoder, and calls her a traitor.

For the particular method used to select the users' codewords, it has to be proved that up to k traitors will not succeed in having an honest user traced in this way, except with negligible probability.

In the terminology of information hiding agreed upon at this workshop (even though it is at its limits in this application), the stegomessages are the personal keys; the embedded messages are the identities or immediately the codewords of the users; there is no real covermessage (the totality of the keys in all buckets comes quite close); and the stegokey consists of all the secret choices the information provider makes and stores.

2.3 Codes, Constructive and Non-Secret Schemes

Because of the way the codewords are chosen, the schemes in [ChFN_94] are mostly non-constructive, i.e., existence is proved, but no explicit construction is given. Furthermore, the more efficient schemes are called secret schemes. This sounds bad, but actually, we can get rid of both these undesirable properties without any real change to the schemes. For instance, we can present the "Secret One-Level Scheme" from [ChFN_94, Section 3.3] as follows:

- For each new user, the information provider randomly and independently chooses a codeword from $\{1, \ldots, b\}^l$ and stores it in a table under this user's identity. The user obtains the corresponding keys.

In [ChFN_94], there is an additional secret mapping from real user identities to other numbers, for which we do not see the need. (A proof of this modified scheme is implicit in Section 3.) Furthermore, Chor, Fiat, and Naor see two disadvantages of such schemes with random codewords over their other, so-called "open" schemes.

The first one is that the honest users' codewords must remain secret from the traitors. However, the full set of all keys from all buckets must remain secret from the traitors anyway, just as almost any cryptographic scheme needs secrets. The scheme would only be "secret" in the undesirable sense if the codewords of the *traitors* had to remain secret from the traitors.

The second disadvantage Chor, Fiat, and Naor see is that there is an error probability, i.e., if the traitors select a codeword among those for which they know the keys, there is sometimes a small chance that this is an honest user's codeword. For this reason, they prefer non-constructive schemes, i.e., schemes of which they only prove existence, where small sets of traitors do not possess a set of keys that would lead to an honest user being traced. However, even in those schemes, the traitors can guess keys they do not possess with a small probability. Thus, it suffices to make the probability of guessing the right codeword equally small as that of guessing the right keys.

3 Making Symmetric Schemes Fully Frameproof

In this section, we modify the scheme described in the previous section so that it becomes fully frameproof, as introduced in Section 1.4.

We replace the decision rule in tracing by the following rule:

- Each user from whom at least l/k keys are found in the pirate decoder is called a traitor. If there is no such user, the output of tracing is *failed*.

Thus the new output *failed* denotes the fact that tracing a traitor failed, because there must have been too many traitors. If at least l/k keys from more than one user are present, the information provider can identify any or all of them as traitors.

Lemma 1. *The symmetric traitor tracing scheme described in Section 2 (in particular Section 2.3) with this new decision rule is*

a) computationally secure for the information provider if there are at most k traitors, and

b) fully frameproof with an exponentially small error probability if the parameters l and b are suitably chosen (as long as the broadcast channel is reliable — a precondition that would also be needed with the original secret one-level scheme in [ChFN_94]).

Proof. The security for the information provider is clear: As before, if up to k traitors have to provide l keys for a pirate decoder, at least one of them must provide at least l/k keys, unless they can find out a key that is not among the personal keys of any of them, which is infeasible if the encryption scheme is secure against known-plaintext attacks.

Now we show that any honest user can only be framed with exponentially small probability. We consider the worst case where all the other users (but not the information provider, because this is a symmetric scheme) collude against her and know all the keys. Recall that the input to the tracing algorithm is one key from each bucket. We can assume without loss of generality that these keys were chosen entirely by the traitors, because the lexicographic selection by the information provider if the pirate decoder contains more keys cannot be worse than a selection by the traitors. Thus, formally the problem is simply: Given the codeword that the traitors have selected, compute the probability that it has at least l/k symbols in common with the honest user's codeword.

The honest user's codeword is chosen randomly and independently, and no information about it becomes known to the traitors. (Here we need the fact that the broadcast channel is reliable, in particular, that honest users only process enabling blocks from the information provider: Otherwise the traitors could send incorrect enabling blocks to the honest user and try to observe at which faults she complains.) The probability that the selected word has l/k symbols in common with a random word can be bounded by using the same version of the Chernoff bound as in [ChFN_94]: Let $X_1, \ldots, X_l$ be independent binary random variables with $Pr(X_i = 1) = p$. Let S denote their sum and E their average, i.e., $S = X_1 + \ldots + X_l$ and $E = S/l$. Then, for all $\beta \geq 1$,

$$Pr(E \geq \beta p) < (e^{\beta-1}/\beta^{\beta})^{pl}.$$

For $\beta = 3$, this implies

$$Pr(E \geq 3p) < 2^{-pl}.$$

In our case, let $X_i = 1$ denote that the traitors guessed the i-th symbol of the honest user's codeword correctly. Thus $p = 1/b$. We want to bound the probability that $S \geq l/k$. If $b \geq 3k$, then

$$Pr(S \geq l/k) = Pr(E \geq 1/k) \leq Pr(E \geq 3/b) < 2^{-l/b}.$$

This is exponentially small, say $\leq 2^{-\sigma}$, if $l \geq \sigma b$, i.e., for instance for

$$b = 3k,\ l = 3\sigma k. \qquad \square$$

For security against active attacks in situations where broadcast is not guaranteed to be reliable, one should at least change the personal key of a user if this user has noticed a wrong enabling block three times.

4 Asymmetric Schemes

First we introduce some notation that is needed for asymmetric traitor tracing.

- We distinguish a *tracing* algorithm and a *trial* protocol.
- Tracing is done by the information provider on his own. For the input to tracing, there is again a choice: Either one simply assumes that a pirate decoder has been opened and information of a certain type has been found inside, or one considers the more general scenario where the pirate decoder is a black box that can be used and reset. The result of tracing is either *failed* or the identity of a user and a string *proof*.
- In the trial protocol, the information provider tries to convince an arbitrary third party, called a *judge*, of the traced user being a traitor. For this, the information provider uses the string *proof* from tracing. Furthermore, the judge needs some input reliably linked to the accused user. We call it the *public key* of this user.

  In some schemes, the accused user has to take part in the trial, others do not need this (in the technical sense). We call such trials 2- and 3-party trials, respectively.
- There is a *key distribution* phase, before or during user initialization, where the user has to distribute a public key reliably (at least so that any future judges will use the correct key, and the same one as the information provider). In all our examples, this will be the key of a digital signature scheme, and it can be used for many other purposes besides the traitor tracing scheme.
- User initialization is now a real protocol where the user also sends at least one message to the information provider.

Now we present several example constructions of asymmetric traitor tracing schemes. More precise definitions could be made similar to [PfSc_96].

4.1 A Simple Scheme with Linear-Size Enabling Blocks

The following simple scheme shows that the idea of asymmetric traitor tracing works for any parameters, i.e., any number of users, any collusion-size, and an arbitrarily small error probability. It is also computationally efficient. Nevertheless, it may not fully count as a broadcast encryption scheme, because the size of the enabling block is linear in N, the number of legitimate users of this session. We call this a *linear-size* scheme, and a scheme where the enabling block is polylogarithmic in N a *log-size* scheme.

Anyway, this scheme could be the method of choice for applications with a small number of users per session, or even a medium number if the information provider fears a large number of traitors, because the efficiency of this scheme does not depend on k. Furthermore, it is natural in applications where individual users ask for permission to decrypt certain data at different times, so that broadcasting enabling blocks is not so suitable anyway.

The scheme works as follows:

- No provider initialization is needed.
- User initialization. The user generates a key pair of an asymmetric encryption scheme, gives the encryption key to the information provider, and signs that the decryption key corresponding to this encryption key will be used exclusively in the traitor tracing scheme. (Recall that many users will already have a key for signing, and the others carry out the key distribution phase as a preamble to user initialization.) The information provider verifies this signature with the public key of this user.
- Session sending. The enabling block contains the session key, encrypted under each encryption key of a legitimate user of this session. This can be seen as a variant of the previous schemes with only one bucket.
- Tracing. The simpler assumption is that a pirate decoder has been opened and the secret decryption key of a traitor has been found inside. The information provider finds out to which encryption key it belongs; if there is no simple predicate to decide this, he can use the ability to decrypt encrypted random messages. If the pirate decoder is a black box with resetting, he uses the same procedure as in Section 2.2 for the one bucket to find out the index j of a key that must be present.
- Trial. The information provider first shows the judge the accused user's signature about the use of her encryption key, and the judge verifies it. Next, if the information provider has actually found a decryption key, he shows it as a proof, and the judge verifies that the de- and encryption keys belong together in the same way as the information provider did. If the information provider has only found the index of a key, he has to hand the pirate decoder to the judge, who verifies that the observed probability of correct decryption for j is smaller than that for $j - 1$ by a certain threshold (smaller than the information provider's threshold).

Lemma 2. *The asymmetric traitor tracing scheme constructed above is computationally secure.*

Proof sketch. First we consider the case where the information provider actually finds a key. The security for the information provider is clear in this case. The security of honest users, i.e., frameproofness against arbitrary collusions even including the information provider, follows from the security of the encryption scheme: It is infeasible to find out any decryption key that belongs to the encryption key of an honest user, and also to find any other way of decrypting encrypted random messages with a significant probability of success. Note that the honest user in the traitor tracing scheme only uses the decryption key to decrypt session keys, which is within the limits for which an encryption scheme should offer security.

In the black-box case, and with a threshold of at most $1/N$, the information provider will certainly identify an index j. If the size of the thresholds and the number of trials are well adapted, the probability that the observation of the judge is below the smaller threshold should be negligible. The security of the honest users, as in the symmetric case, relies on the fact that it is infeasible to distinguish encryptions of a known share from encryptions of random values from the message space. □

Note that in the black-box case, we have used strong cryptographic properties of the encryption scheme. In particular, it would be insecure with deterministic encryption.

4.2 Schemes with Shorter Enabling Blocks

Now we show an approach to turn a given symmetric scheme whose enabling blocks are shorter than linear in the number of legitimate users into an asymmetric scheme. In this section, we only obtain an instantiation for $k = 1$, like the very first symmetric scheme in [ChFN_94]. In Section 4.3, we present a variation of the construction that allows $k > 1$ at the cost of an error probability that is not exponentially small.

The construction is similar to that of asymmetric fingerprinting in [PfSc_96]. The following assumptions have to be made about the way the users' codewords are chosen in the given symmetric scheme:

1. The input to the tracing algorithm is only the information found in a pirate decoder and the information provider's initial record, i.e., no records from individual user initializations are needed.
2. The identity space is nevertheless large enough to serve as the preimage of a one-way function.

We first show that the scheme for $k = 1$ at the beginning of [ChFN_94, Section 3.1] is of this form: A user's codeword is simply this user's identity. (Note that for $k = 1$, codewords need not be secret, because one traitor alone has no other keys than exactly her own ones to put into a pirate decoder.) The tracing algorithm, upon finding one traitor's personal key, and thus this traitor's codeword, immediately has the identity. The identity space can be made exponentially large, because the scheme is polynomial in the length of the identities.

For the following construction, two cryptographic primitives must be fixed. One is a one-way function f. The other is a protocol for secure 2-party computation. Such a protocol achieves the following goal: Two parties have secret inputs x_1 and x_2, respectively. Both or one of them want to know $g(x_1, x_2)$, where g is a function known to both of them. However, Party 1 should not gain additional information about x_2, nor Party 2 about x_1. A nice example is the millionaires' problem from [Yao_82]: The parties are millionaires, and x_1 and x_2 denote how many millions they own. They want to find out who is richer, but without telling each other x_1 and x_2. Constructions have been given, e.g., in [Yao_86, ChDG_88, GMW_87].

Given the one-way function and the 2-party protocol, the asymmetric traitor tracing scheme is constructed as follows:

- Provider initialization is the same as in the symmetric scheme.

- User initialization works as follows: The user first chooses a random “identity” id_{sym} that remains secret from the information provider. Then the 2-party computation protocol is used for the following secret computation:
  - The user secretly inputs id_{sym} and her secret signing key.
  - The information provider secretly inputs the $l \cdot b$ secret keys from provider initialization and (not necessarily secretly) the user’s public test key.
  - The one-way image $f(id_{sym})$ and a signature *sig* under it (and a suitable explanatory text) with the user’s secret key are computed. The signature is internally verified with the test key input by the provider. If it is wrong, e.g., because the user has input an incorrect secret key, the protocol is aborted. Then the codeword and thus the personal key corresponding to id_{sym} are computed.
  - First, $f(id_{sym})$ and *sig* are output to the information provider, who stores them under the real identity of this user. Secondly, the personal key is output to the user.
- In tracing, the information provider uses the symmetric tracing algorithm (black-box or not) to find out id_{sym}. He applies f to it and compares the result with the stored values $f(id_{sym})$ to find out the real identity of the user who used this id_{sym}.
- In a trial, the information provider uses id_{sym} and *sig* as a proof. The judge verifies that *sig* is a signature by the accused user on $f(id_{sym})$ and the explanatory text.

Lemma 3. *Let a secure symmetric traitor tracing scheme fulfilling Conditions 1 and 2, a secure one-way function, and a secure 2-party protocol (in the sense that it can replace a trusted oracle computing the same function) be given. Then the asymmetric traitor tracing scheme constructed above is computationally secure.*

This holds for the same reasons as the security of Construction 1 in [PfSc_96]. We omit a proof sketch, but present a related one in Section 4.3.

As to efficiency, note that the 2-party protocol only has to be carried out once in user initialization and does not affect the transmission of the actual broadcast data. Nevertheless, the scheme is rather theoretical as long as a general 2-party protocol has to be used. One can certainly try to improve the efficiency by exploiting the specific structure of the function computed here; in particular, the selection of the keys is a kind of all-or-nothing disclosure of secrets.

4.3 3-Party Trials for Small Codes

We now consider the case where we cannot fulfil the conditions of Section 4.2 and present schemes that are not fully secure asymmetric schemes, but where an honest user can at least fend off a false accusation with a certain probability, such as $1-2^{-10}$. Such a scheme makes both treachery and false accusations quite risky (under the assumption that false accusations will at least decrease the reputation of the information provider and may even be punished). If the underlying symmetric scheme is fully frameproof, an honest information provider is protected from making false accusations and thus endangering his reputation.

Thus we use the symmetric scheme that was considered in Lemma 1. To be slightly more general, we represent it as follows:

- In provider initialization, the information provider already makes all the random choices that he may need later. This may simply mean generating a sufficiently long string of random bits for later use. The result of provider initialization is still called the initial record, *record* = (*keys*, *code_randomness*), where *keys* is the matrix of all the keys in all the buckets.

- In user initialization, the information provider assigns a number $i \in \{1, \ldots, N\}$ to the user and computes the user's codeword as *code*(*code_randomness*, *i*), where *code* is a certain globally known, deterministic algorithm.
- The decision rule in tracing remains the same, and we also still presuppose that full frameproofness would be guaranteed even if the traitors could choose their own codewords. (We can omit this last condition by choosing the value *evid* within the 2-party protocol in the construction below.)

In this notation, the scheme where codewords are chosen randomly looks like this: *code_randomness* is a table that assigns a random codeword to each user number $i \in \{1, \ldots, N\}$, and *code* is the fixed table-lookup function. One can easily see from the proof of Lemma 1 that the last condition holds in this case. In a scheme with a deterministic code, *code_randomness* would be empty, and *code* would be the code in the usual sense.

The idea for constructing an asymmetric scheme from such a symmetric scheme is similar to Section 4.2, but instead of hiding the whole identity computationally, we now hide a few bits information-theoretically. Thus, instead of a one-way function, we use commitments. Commitments are the cryptographic analog of the following protocol: In the *commit* phase, one party puts some bits in a wooden box, locks the box, and sends it to the other party or puts it in a public place. Now the bits are hidden, but they can no longer be changed. In the *reveal* phase, the party hands over the key of the box, so that the recipients can verify what bits the box contains. More technically, the commit phase has two outputs: the commitment, *com*, is an output for both parties, and the committing party additionally obtains a string *reveal_info*, which later enables her to carry out the reveal phase. For several cryptographic realizations, see [BrCC_88]. Among them, we need one where the bits are information-theoretically hidden, whereas the fact that the bits cannot be changed after the commit phase need only hold computationally.

For the asymmetric traitor tracing scheme, the identities of the symmetric scheme are divided into two substrings. The first one will be called the *tracing identity*, the second one the *evidence of treachery*. For instance, $N = 10^9$ was mentioned as realistic in [ChFN_94]. We can use this for 20-bit tracing identities and 10-bit evidence of treachery, which means that about 10^6 users can take part, and that they can fend off false accusations with a probability of about $1 - 10^{-3}$.

As in Section 4.2, each user must have published the public key of a signature scheme, and we need a protocol for secure 2-party computation.

- In user initialization, a user is assigned a tracing identity id_{trace} by the information provider. The user secretly chooses the evidence of treachery, *evid*. Now the 2-party protocol is used:
  - The information provider secretly inputs *record* = (*keys*, *code_randomness*) and (not necessarily secretly) id_{trace} and the public key of this user.
  - The user secretly inputs *evid* and her secret signing key.

  - A commitment *com* to *evid* is made, and a signature *sig* on *com* (and a suitable explanatory text) is computed with the user's secret key. The signature is verified with the user's public key. If this fails, the protocol is aborted. Otherwise, the concatenation of id_{trace} and *evid* is called id_{sym}, and the user's codeword *code*(*code_randomness*, id_{sym}) and the corresponding personal key are computed.
  - First, the string *reveal_info* needed to open the commitment is output to the user, then *com* and *sig* are output to the information provider, and finally the personal key is output to the user.
- In tracing, the information provider uses the symmetric tracing algorithm (black-box or not) to find a word of indices of keys used. He searches for a codeword, i.e., a value in the range of *code*(*code_randomness*, •), that equals the given word in at least l/k positions. If none is found, the output is *failed*. Otherwise, he splits the corresponding value id_{sym} into id_{trace} and *evid*. He looks up the real identity of the user who was assigned id_{trace} and the corresponding values *com* and *sig*.
- In the trial, the information provider uses the triple (*evid*, *com*, *sig*) as a proof. The judge first verifies that *sig* is a signature on *com* and the appropriate text by the accused user. Intuitively, the information provider's claim is now that *evid* is the content of *com*. To fend off such an accusation, the user has to prove that she can open her commitment in a different way. This can simply be done by showing the real *evid* and *reveal_info*, if the user can never be accused about this commitment again. Otherwise, a zero-knowledge proof has to be used. (This is at least no problem if the reveal phase is non-interactive, which is the case in most realizations.)

Lemma 4. *If the symmetric traitor tracing scheme is fully frameproof and the commitment scheme and the 2-party protocol used are secure (the latter in the sense that it can replace a trusted oracle computing the same function), the asymmetric traitor tracing scheme just constructed is secure, except that the probability that an honest user cannot fend off a wrong accusation is not exponentially small, but of the order* $2^{-\sigma}$*, where* σ *is the length of evid.*

Proof sketch. a) First we consider an honest user. Assuming that the 2-party protocol has been replaced by a perfect oracle, the only information about *evid* that becomes known to anybody else is (*com*, *sig*). By the security of the commitment scheme, this means that any accusation has to be based on a random guess at *evid*, which will only be correct with the probability $2^{-\sigma}$. If this guess is wrong, the user can fend off the accusation: The judge in a trial only accepts a value *com* signed by this user with the correct explanatory text, i.e., the information provider has to use the correct commitment, and the user can open it to reveal the correct *evid*. (This is why *reveal_info* had to be output first.)

b) Next we show that even an arbitrary number of traitors cannot trick the information provider into making false accusations, except with exponentially small probability: The traitors have to choose a word to put into the pirate decoder. By the full frameproofness of the symmetric scheme (even if the traitors can choose *evid*, by our preconditions), with very high probability, this word does not have l/k symbols in common with any honest user's word. Thus the information provider will not accuse an honest user.

c) Finally, we show that it is infeasible for at most k traitors to cheat the information provider. As in the symmetric scheme, at least one traitor has to provide at least l/k keys. Thus the word found in tracing has l/k positions in common with the codeword of a traitor.

Furthermore, by Part b) it does not have *l/k* positions in common with any honest user's codeword (except with negligible probability). Thus the information provider identifies the codeword of a traitor. By the security of the 2-party protocol, the corresponding value id_{sym} = (id_{trace}, *evid*) was used in user initialization with this traitor, and *evid* is the content of this traitor's commitment. By the security of the commitment scheme, the traitor cannot prove that she can open the commitment differently, and thus cannot fend off the accusation. □

5 Summary and Outlook

We have presented improvements to traitor tracing schemes for broadcast encryption, in particular to the possibility of trials based on them.

Primarily, we introduced asymmetric traitor tracing, which allows responsibility for redistributed information to be assigned to one particular person, and we presented example constructions. Only one special case, though, is already efficient enough for practical use. In future work, we will construct more efficient schemes with 2-party trials from the two-level schemes of [ChFN_94] and more general schemes with 3-party trials.

We also presented efficient improvements to symmetric traitor tracing, in particular fully frameproof schemes, where no collusion of users can frame another user as a traitor. Furthermore, we discussed the general restrictions of traitor tracing that must be taken into account when choosing between traitor tracing and fingerprinting of the actual data.

Acknowledgments

I am happy to thank *Matthias Schunter* and *Michael Waidner* for helpful comments on a draft of this paper, and the Isaac Newton Institute in Cambridge for hosting me during a revision.

References

BDHK_93 Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, Moti Yung: Perfectly-Secure Key Distribution for Dynamic Conferences; Crypto '92, LNCS 740, Springer-Verlag, Berlin 1993, 471-486.

Berk_91 Shimshon Berkovits: How To Broadcast A Secret; Eurocrypt '91, LNCS 547, Springer-Verlag, Berlin 1991, 535-541.

BlCr_95 Carlo Blundo, Antonella Cresti: Space Requirements for Broadcast Encryption; Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 287-298.

BlMP_86 G. R. Blakley, Catherine Meadows, G. B. Purdy: Fingerprinting Long Forgiving Messages; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 180-189.

Blom_80 Rolf Blom: Key Distribution for Broadcast Cryptography; Third International Conference: Security Through Science and Engineering, September 23-26, 1980, Technical University Berlin, 19-23.

BoSh_95 Dan Boneh, James Shaw: Collusion-Secure Fingerprinting for Digital Data; Crypto '95, LNCS 963, Springer-Verlag, Berlin 1995, 452-465.

BrCC_88 Gilles Brassard, David Chaum, Claude Crépeau: Minimum Disclosure Proofs of Knowledge; Journal of Computer and System Sciences 37 (1988) 156-189.

Caro_95 Germano Caronni: Assuring Ownership Rights for Digital Images; Proceedings VIS '95, Vieweg, Wiesbaden 1995, 251-263.

ChDG_88 David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

ChFN_94 Benny Chor, Amos Fiat, Moni Naor: Tracing traitors; Crypto '94, LNCS 839, Springer-Verlag, Berlin 1994, 257-270.

DwLN_96 Cynthia Dwork, Jeffrey Lotspiech, Moni Naor: Digital Signets: Self-Enforcing Protection of Digital Information; to appear at 28th Symposium on Theory of Computing (STOC) 1996; preliminary version received April 1996.

FiNa_94 Amos Fiat, Moni Naor: Broadcast Encryption; Crypto '93, LNCS 773, Springer-Verlag, Berlin 1994, 480-491.

GMW_87 Oded Goldreich, Silvio Micali, Avi Wigderson: How to play any mental game – or – a completeness theorem for protocols with honest majority; 19th Symposium on Theory of Computing (STOC) 1987, ACM, New York 1987, 218-229.

PfSc_96 Birgit Pfitzmann, Matthias Schunter: Asymmetric Fingerprinting; Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 84-95.

Wagn_83 Neal R. Wagner: Fingerprinting; Proceedings 1983 IEEE Symposium on Security and Privacy, April 25-27 1983, Oakland, California, 18-22.

Yao_82 Andrew C. Yao: Protocols for Secure Computations; 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 160-164.

Yao_86 Andrew C. Yao: How to Generate and Exchange Secrets; 27th Symposium on Foundations of Computer Science (FOCS) 1986, IEEE Computer Society, 1986, 162-167.

ZhKo_95 Jian Zhao, Eckhard Koch: Embedding Robust Labels Into Images For Copyright Protection; Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, R. Oldenbourg Verlag, München 1995 (also at http://www.igd.fhg.de/www/igd-a8/pub/EmbedLabel.ps).