Threshold Traitor Tracing

Moni Naor* and Benny Pinkas**

Dept. of Applied Mathematics and Computer Science
Weizmann Institute of Science
Rehovot 76100, Israel

Abstract. This work presents *threshold tracing schemes*. Tracing schemes trace the source of keys which are used in pirate decoders for sensitive or proprietary data (such as pay-TV programs). Previous tracing schemes were designed to operate against any decoder which decrypts with a *non-negligible* success probability. We introduce threshold tracing schemes which are only designed to trace the source of keys of decoders which decrypt with probability greater than some threshold q (which is a parameter). These schemes present a dramatic reduction in the overhead compared to the previous constructions of tracing schemes.
We argue that in many applications it is only required to protect against pirate decoders which have a decryption probability very close to 1 (for example, TV decoders). In such applications it is therefore very favorable to use threshold tracing schemes.

1 Introduction

We present very efficient tracing systems: systems which allow data providers to identify sources of leakage of their keys to illegitimate receivers. Consider for example a pay-TV provider which finds out that someone is selling pirate decoders which enable the decoding of transmissions without paying the required fees. A tracing system enables the provider to identify which legitimate receivers assisted in constructing the pirate decoders.

Tracing systems were first presented by Chor, Fiat and Naor [8]. They used the following security requirement, which in our view is too stern for many applications: they required full-resiliency, i.e that the schemes should trace the source of any decoder which decodes with a *non-negligible* probability. We claim that for many very relevant applications a decoder with a success probability which is non-negligible, but is not very close to 1, is useless. Assume for example that a TV program is divided into one minute segments which are separately encrypted. A decoder which decrypts with probability 90% is expected to fail in the decoding of one out of ten minutes. Very few customers will be willing to pay for such a decoder.

* Research supported by BSF Grant 32-00032. E-mail: naor@wisdom.weizmann.ac.il.
** Supported by an Eshkol Fellowship from the Israeli Ministry of Science. E-mail: bennyp@wisdom.weizmann.ac.il.

We present *threshold tracing schemes* which depend on a parameter q. They trace the source of keys of any decoder which decodes with success probability not smaller than q but there is no guarantee for their success against decoders with success probability smaller than q. The efficiency of our threshold tracing schemes is superior to that of the tracing schemes of [8] (see Section 4.3 for a numerical comparison for constructions of typical size). We therefore claim that applications which do not require fully resilient tracing should use threshold tracing schemes.

In order to use threshold tracing schemes the communicated content should be divided into blocks which are independently encrypted. A legitimate decoder contains keys which enable it to decrypt every block. These keys identify that decoder. If a (*pirate*) decoder contains enough keys (taken from the legitimate decoders of traitors) to enable it to decrypt more than a q fraction of the blocks, these keys are sufficient to identify at least one of the traitors. It is assumed that a pirate decoder which decrypts less than a q fraction of the blocks is not useful and therefore it is not important to trace the source of its keys.

In general, it is always useful to recognize what is a "success" of the adversary, and design schemes which prevent such a success. This process may lead to very efficient constructions, with an overhead that is proportional to the severity of the "attack" to which they are immune (this is the case with the threshold tracing schemes we present, whose overhead is an inverse function of q). Such constructions can also serve to price the security by presenting the overhead incurred by requiring a certain amount of security.

Let us first consider the scenario in which the schemes operate. A *data provider* is distributing some content to legitimate receivers (e.g. paying subscribers). The content is typically distributed encrypted, and each legitimate receiver has a decryption key. A *traitor* is a legitimate receiver who attempts to enable unauthorized users to access the content. A traitor can distribute a copy of the cleartext of the content to other illegitimate receivers. We do not attempt to protect against such pirate distribution but claim that in many cases the economy of scale makes such a distribution non-profitable or too dangerous. Typical cases where this is true include

- Pay-per-view or subscription television broadcasts. It is an expensive and a risky business to start a pirate broadcast station. (A similar application is the distribution of content over the Internet using "push" technology).
- Online services or databases, publicly accessible (say on the Internet) where a charge may be levied for access to all or certain records. The pirate must copy the entire information provided by the online service and maintain an updated copy. This process is non-efficient and can be easily detected.

As piracy in these cases is a criminal commercial enterprise the risk/benefit ratio in distributing illegal copies of the content becomes unattractive. A pirate can sell illegal access to the content by providing its customers with much shorter data – the decryption keys. We therefore concentrate in this paper in preventing traitors from distributing their decryption keys to other users[1]. We construct *(k,q)-threshold tracing schemes*. If an illegitimate user uses a pirate decoder[2] which was built using the keys of at most k legitimate users (who are therefore traitors), and if *the decoder can decrypt with probability at least* q, then our schemes will identify (with high probability) at least one traitor given the pirate decoder (it cannot be promised that the schemes identify more traitors since it is possible that all the keys used in constructing the pirate decoder were taken from a single traitor).

We note that in fact our schemes have the very desirable property that the identity of the traitor can be established by considering the pirate decryption process as a *black box*. It suffices to capture one pirate decoder and its behavior will identify the traitor, there is no need to "break it open" or read any data stored inside.

The schemes can be based on any symmetric encryption system. The security parameter is the length of the key of that system. We measure the *efficiency* of the solutions in terms of several performance parameters. The memory and communication parameters are measured in multiples of the size of the security parameter. The efficiency parameters are:

(a) The memory and computation requirements for an authorized user. These parameters are of special importance if the user has limited computation and storage capabilities, as is the case with smartcards.
(b) The memory and computation requirements for the data supplier. These parameters are typically less important, since the data supplier can perform its computations off-line and can use large storage space.
(c) The data redundancy overhead, i.e. the increase in data size that is needed in order to enable the tracing. This refers to the communication overhead (in broadcast or online systems) or the additional "wasted" storage in CD-ROM type systems.

1.1 Our Results

Consider a tracing scheme for n users, which should be secure with probability $1-p$ against coalitions of up to k users.

[1] In practice today it is often considered sufficient to prevent piracy by supplying the authorized parties with so-called secure hardware solutions (smartcards and their like) that are designed to prevent interference and access to enclosed cryptographic keys. The assumptions about the security of these hardware mechanisms are not always correct. There are several methods that use hardware faults in the "secure hardware solutions" in order to find the keys that are enclosed inside [3, 5, 4]. Our schemes obtain their claimed security without any secure hardware requirements. Should such devices be used to store the keys, they will undoubtedly make the attack even more expensive, but this is not a requirement.

[2] We use the term pirate decoder to represent the pirate decryption process, this may or may not be a physical box, and may simply be some code on a computer.

Our schemes compare very well to the the best tracing scheme of [8] (see also table 1). That scheme required each user to store a personal key of length $O(\log(1/p)\log(n/p))$. This was also the running time required from the user. The communication overhead was $O(k\log(1/p)\log(n/p))$. We remark that the "O" notation hides considerable coefficients.

For a threshold $0 < q < 1$, our one-level scheme has personal keys of length $\frac{4k}{3q}\log(n/p)$, and a communication overhead of only $4k$. The user is required to perform only a single decryption operation.

The length of the personal keys of the simplest two-level threshold scheme is $O(\log(k/p)\log(n/p))$, and its communication overhead is $O(k\log(\frac{k}{q\log(k/p)}))$. A user should perform very few decryption operations. We remark that in this case the coefficients of the "O" notation are very moderate. Table 1 contains a comparison for a reasonable size system, in which all the parameters and coefficients are plugged in

From now on we describe the exact complexity of the schemes we present. We do not use an "O" notation but rather present all the constant coefficients.

1.2 Content Distribution Schemes

The schemes which are used to distribute the content from the data provider to the legitimate receivers are of the following general form: The data supplier generates a *meta-key* which contains a base set A of random keys and assigns subsets of these keys to users, m keys per user (the parameters will be specified later). These m keys jointly form the user's *personal key*. Different personal keys may have a nonempty intersection. We denote the personal key for user u by $P(u)$, which is a subset of the base set A.

A message in a traitor tracing message consists of pairs of the form $\langle$*enabling block, cipher block*$\rangle$. The cipher block is the symmetric encryption of the actual data (say part of a video clip), under some secret random key s. Alternately, it could be the exclusive-or of the message with s and we would get an information theoretic secure version of the scheme (although a very inefficient one, since as with any one-time-pad the size of the key should be as long as the encrypted data). The enabling block allows authorized users to obtain s. The enabling block consists of encrypted values under some or all of the keys of the base set A. Every authorized user will be able to compute s by decrypting the values for which he has keys and then computing the actual key from these values. For all the schemes we present the computation on the user end is simply taking the exclusive-or of values that the user is able to decrypt.

A very simple scheme is to give each user a different key. Then the enabling block includes an encryption of s with each of the users' keys. However the length of the enabling block is then linear in the number of legitimate users and might be too large for many applications.

Traitors may conspire and give an unauthorized user (or users) a subset of their keys so that the unauthorized user will also be able to compute the key s from the values he has been able to decrypt. The goal of the system designer is to assign keys to the users such that when a pirate decoder is captured it would be possible to detect at least one traitor, subject to the limitation that the number of traitors is at most k. We remark that the overhead of both the schemes of [8] and of our threshold schemes depends on the parameter k. Since the overhead of our schemes is a considerably smaller function of k it is possible to set this parameter to a higher value and protect against larger coalitions.

1.3 Eliminating Piracy

Traitor tracing schemes help in three aspects of piracy prevention: they deter users from cooperating with pirates, they identify the pirates and enable to take legal actions against them, and they can be used to disable active pirate users.

The usage of traitor tracing schemes discourages users from helping pirates and especially from submitting their keys to be used in pirate decoders. In particular, if the process of a user obtaining a personal key requires some sort of registration and physical identification then it should be hard for pirates to obtain a large number of personal keys. Consequently, the tracing traitor scheme can identify the source of keys which are used in pirate decoders and this mere fact should deter users from helping pirates.

When a pirate decoder is found and a source of its keys is identified, legal activities should be taken against this source. Indeed, as was pointed by Pfitzmann in [17] a corrupt data provider that wishes to incriminate an honest user might construct a "dummy" pirate decoder containing this user's keys, "reveal" it and claim that the user is a pirate. Similar misbehavior is possible though with many current types of services and yet there is little evidence that service providers have performed such illegal activities.

The broadcast encryption schemes of Fiat and Naor [13] deal very efficiently with disabling active pirate users, i.e. preventing them from further decryption. These schemes allow one to broadcast messages to any *dynamic* subset of the user set and are specifically suitable for pay-per-view TV applications. The schemes require a single short transmission to disable all pirate decoders if they were manufactured via a collaborative effort of no more than k traitors. Another broadcast encryption scheme was suggested by Wallner et al [18], and is secure against any number of corrupt users. It has better performance than [13] if the number of deletions is small. In particular, personal keys are of length $O(\log n)$ and there is no data redundancy in regular operation.

A combination of a traitor tracing scheme and a broadcast encryption scheme is a very powerful tool. When a traitor is traced the dynamic subset of users authorized to receive the broadcast should be changed by simply excluding the traced traitor. This procedure should be repeated until the pirate box is rendered useless. In [9] it is described how to combine a tracing traitor scheme and a broadcast encryption scheme in order to achieve this capability. Both the data redundancy overhead and the key length of the resulting scheme are the multiplication of the corresponding overheads for the tracing and broadcast encryption schemes (but used with the scheme of [18] this does not increase the total overhead too much).

1.4 Related Work

The work of Chor, Fiat, and Naor [8] has introduced the concept of traitor tracing, and presented several tracing schemes. We survey their results in section 3. A more complete and formal treatment of the problem is presented in [9] which is the full version of [8] and of our paper.

Boneh and Shaw [6] have suggested a scheme for marking different copies of an electronic document by inserting a different fingerprint into each copy. The fingerprint is composed of a sequence of marks with each mark having one of two values (therefore the fingerprint corresponds to a binary string)[3]. The scheme is based on a marking assumption which states that a coalition of users who each have a copy with the same value for a certain mark cannot generate a copy with a different value for that mark. The scheme has the property that using up to k copies it is impossible to generate a new copy whose fingerprint does not reveal at least one of the k copies that were used. It offers better security in the sense that it enables to trace the leaked content itself (and not just the key which enables its decryption). It can also be used as a tracing traitors scheme but it is much less efficient than the schemes of [8]: the number of keys that each user should have is k^4 times greater than in the most efficient scheme of [8].

Another solution for copyright protection is through *self enforcement* schemes, which were suggested by Dwork, Lotspiech and Naor [11]. In these schemes the content is encrypted and each legitimate user receives a different decryption key which includes some sensitive information related to the user (e.g. his credit card number). Users will be reluctant to hand their keys to others since the keys contain this sensitive information. The self enforcement schemes suggested in [11] use the same type of security as was used in [8, 6]. Namely, the system is secure against coalitions of less than k corrupt users, and the system's complexity depends on k.

Pfitzmann [17] has suggested a tracing traitors method which yields a proof for the liability of the traced traitors. In this scheme the issuing of keys from the center to the users is performed by an interactive protocol. At the end of the protocol the center is not able to construct a "pirate decoder" that frames a user, but if a real pirate decoder is found the center is able to trace the source of the keys the decoder contains. However, as this construction uses a relatively complex primitive (general secure multi party protocols) which is rather inefficient (e.g. it operates on the circuit which evaluates the function), its overall complexity is high.

2 Definitions

A *traitor tracing* scheme consists of three components:

- A *user initialization scheme*, used by the data supplier to add new users. The data supplier has a meta-key α that defines a mapping $P_\alpha : U \mapsto \{0,1\}^s$

[3] See for instance [10] for a method for inserting marks into a document.

where U is the set of possible users and s is the number of bits in the personal key of each user. When user $u_i \in U$ joins, he receives his personal key $P_\alpha(u_i)$. In all of our constructions $P_\alpha(u_i)$ consists of a subset of m decryption keys out of a larger set A of keys.
– An *encryption scheme* $E_\alpha : \{0,1\}^* \mapsto \{0,1\}^*$ used by the data supplier to encrypt messages and a *decryption scheme* $D_\beta : \{0,1\}^* \mapsto \{0,1\}^*$ used by every user to decrypt those messages. Let the personal key of user u_i be $\beta = P_\alpha(u_i)$, then for any message $M \in \{0,1\}^*$ we have $M = D_\beta(E_\alpha(M))$. In our schemes the messages are encrypted block by block where every encrypted block contains an enabling block and a cipher block. The decryption process consists of a preliminary decryption of encrypted keys in the enabling block, a process which combines the results to obtain a common key, and finally a decryption of the cipher block.
– A *traitor tracing algorithm*, used upon confiscation of a pirate decoder, to determine the identity of a traitor. We do not assume that the contents of a pirate decoder can be viewed by the traitor tracing algorithm but rather that the tracing algorithm can access it as a black box and test how (if at all) it decrypts an input ciphertext. (We do assume however that the pirate decoder can be reset to its original state, i.e. that there is no self-destruction mechanism when it detects a traitor tracing algorithm.)

The encryption of plaintext blocks in our schemes results in a message which consists of an *enabling block* and a *cipher block*. The cipher block contains the plaintext block encrypted by some encryption algorithm using some random *block key* s which is unique to this block. The enabling block contains encryptions of "shares" of the block key such that every legitimate user can use his personal key to decrypt enough shares to reconstruct the block key. An adversary who wants to decrypt the message can either break the encryption scheme that was used in the cipher block without using any information from the enabling block, or try to learn some information from the enabling block that might help in the decryption process. In this paper we assume that it is hard to break the underlying encryption scheme so we are only interested in preventing attacks of the latter kind.

Assume that an adversary has the cooperation of a coalition of at most k legitimate users, and uses their keys to construct a decoder. We would like to trace at least one of the coalition members. Intuitively a scheme is called *fully resilient* if it is possible to trace (with high certainty) at least one of the traitors that helped build a decoder which does not break the underlying encryption algorithms. More accurately, a system is fully resilient if for every pirate decoder which runs in time t it either holds that it is possible to trace at least one of the traitors which helped its construction, or that the decoder can break one of the underlying encryption algorithms in time t.

Fully resilient tracing schemes were suggested and constructed in [8]. There are many applications for which the pirate decoder must decrypt with probability close to 1, like the TV broadcast example we presented in Section 1. In such scenarios we can concentrate on tracing the source of keys which were used to build decoders which decrypt with probability greater than some threshold. A scheme is called a *q-threshold scheme* if for every decoder which does not break the underlying encryption algorithms and decrypts with probability greater than q it is possible to trace at least one of the traitors that helped building it.

An obvious and preliminary requirement from the tracing traitors schemes is that they supply secure encryption. That is, an adversary which has no information on the keys that are used should not be able to decrypt the encrypted content. Intuitively, our security definitions claim that if an adversary (who might have some of the keys) is able to decrypt and escape from being traced then the scheme is insecure as an encryption scheme *even against an adversary who has no keys.*

Following we present an exact definition of fully-resilient and threshold tracing schemes.

Definition 1. Let T be a coalition of at most k users. Let $\mathcal{A}$ be an adversary who has a subset F of the values of the keys of the users in T, and who is able to decrypt in time $\mathbf{t}$ and with probability greater than $\mathbf{q}'$ the content sent in the tracing traitors scheme.

The security assumption is that one of the following two statements holds:

- Given F the data supplier is able to trace with probability at least $\mathbf{1} - \mathbf{p}$ at least one of the users in T.
- There exists an adversary $\mathcal{A}'$ which uses $\mathcal{A}$ as a black box and whose input is only an enabling block and a cipher block of the tracing traitors scheme. $\mathcal{A}'$ can reveal the content that is encrypted in the cipher block in time which is linear in the length of its input and in $\mathbf{t}$, and with probability at least $\mathbf{q}''$ (q'' is defined in the next paragraph).

The probability is taken over the random choices of the data supplier, and when appropriate over the random choices of the adversary or of the tracing algorithm.

The scheme is called fully (p, k)-resilient if the security assumption holds for $q' = q''$. If the scheme further achieves $p = 0$ then it is called fully k-resilient.

The scheme is called q-threshold (p, k)-resilient if the security assumption holds for $q' = q + q''$.

Since we assume the underlying encryption algorithms to be secure, we can assume that the probability (q'') with which an adversary $\mathcal{A}'$ which knows nothing but the ciphertext can break the encryption is negligible. Therefore in a fully resilient scheme the data supplier can trace at least one traitor if it finds a pirate decoder (adversary $\mathcal{A}$) which decrypts with non-negligible probability. In a threshold scheme the data supplier is able to do so if it finds a decoder which decrypts with probability greater than q by a non-negligible difference (but to simplify the exposition we often take the freedom to refer to threshold schemes as secure against any pirate decoder which decrypts with probability greater than q).

3 Fully-Resilient Tracing Schemes

The fully-resilient tracing schemes of [8] are based on applying hash functions combined with any private key cryptosystems, and do not require any public key operations. Our threshold schemes will be based on the same operations. The hash functions are used to assign decryption keys (from a base set of decryption keys) to authorized users. The assignment guarantees that any combination of keys, taken from the personal keys of any coalition of traitors, has the following property: If this combination enables decryption then it is "far" from the personal key of any innocent (non–traitor) user. (For more information on hash functions and their applications see [15, 7, 19, 14].)

There are two types of traceability schemes defined in [8]. *Open schemes* assume that the mapping is public and the indexes of the keys which are mapped to any user are publicly known (the only secret information is the content of the keys). *Secret schemes* are defined to operate in cases where the mapping of keys is secret and it is unknown which keys are used by every user. The constructions of secret schemes can be more efficient than those of open schemes and are therefore recommended to be used in practice. The reason for the gain in efficiency is that traitors do not know which keys the other users received. Therefore even if the set of keys of a coalition of traitors includes a large part of the keys of an innocent user the traitors do not know which keys these are and cannot construct a pirate decoder which incriminates a specific user. Our threshold tracing schemes are secret schemes.

Secret fully-resilient schemes were constructed for n users and at most k traitors. Two types of secret schemes were presented in [8]:

- Secret fully (p, k)-resilient one-level schemes required the personal key of each user to consist of $m = \frac{4}{3}k\log(n/p)$ decryption keys, and the enabling block to include $\frac{16}{3}k^2\log(n/p)$ key encryptions. Each user should perform $\frac{4}{3}k\log(n/p)$ decryptions in order to reveal the broadcasted secret.
- Secret fully (p, k)-resilient two-level schemes required the personal key of each user to consist of $m = \frac{4}{3}b\log(2n/p)$ decryption keys, and the enabling block to include $\frac{32}{3}ekb\log(2n/p)(1+\frac{\ln(ek/b)}{b-1-\ln(ek/b)})$ key encryptions, where $b = \log(4/p)$. Each user should perform $\frac{4}{3}b\log(2n/p)$ decryptions in order to reveal the broadcasted secret. Two-level schemes are more efficient than one-level schemes if $k \gg \log(1/p)$.

4 Threshold Tracing Schemes

Threshold tracing schemes are designed to trace the source of keys of any pirate decoder whose advantage in decrypting the content (compared to an adversary who does not have any of the keys) is at least q.

The complexity of q-threshold schemes depends on q. These schemes are more efficient for larger values of q. They are secret schemes in the sense that the set of keys that each user receives is unknown to other users. The design concept of these schemes is as follows: *either the pirate decoder holds enough keys to enable the tracing of at least one traitor*, or *it does not contain enough keys to ensure a decryption probability greater than q*. The security of tracing schemes is reduced to the assumption that the encryption scheme that is used is secure and therefore any adversary who does not have the decryption keys cannot decrypt with a non-negligible success probability.

The benefit of using threshold tracing schemes is a dramatic reduction in the data redundancy overhead and in the number of operations needed for decryption, whereas the length of the personal key is almost as short as in secret fully resilient schemes. We also present a threshold scheme which improves over fully-resilient schemes in all complexity parameters. Next we define one-level and two-level threshold tracing schemes. The data redundancy overhead and the personal key length are parameterized and there is a tradeoff between them. It is possible to set the parameter to a value which obtains the best tradeoff between the two complexity measures (for instance the last entry of Table 1 demonstrates a reasonable such tradeoff).

4.1 A One-Level Threshold Scheme

The scheme uses a threshold parameter q, against k traitors and for a total of n users, each with a unique identity $u \in \{1, \ldots, n\}$.

Initialization: A set of ℓ hash functions $h_1, h_2, \ldots, h_\ell$ are chosen independently at random. Each hash function h_i maps $\{1, \ldots, n\}$ into a set of $4k$ random keys $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,4k}\}$. The hash functions are kept secret. User u receives, upon initialization, the indices and values of ℓ keys $\{h_1(u), h_2(u), \ldots, h_\ell(u)\}$. The keys can be imagined as organized in a matrix of size $\ell \times 4k$, where each user receives a single key from each row.

Distributing a secret: Let s be the secret to be distributed. Let $q \leq w < 1$ and $0 < t \leq \ell$ be two parameters which will be defined later (the scheme would divide the secret into t shares and ensure that a decoder which contain keys from fraction of at least w of the rows would be able to decrypt the secret with probability greater than q).

The data provider chooses random values $\{s_i\}_{i=1}^{t}$ subject to the constraint $\oplus_{i=1}^{t} s_i = s$, and chooses t random rows $r_1, \ldots, r_t$. For every i $(i = 1, 2, \ldots, t)$ the data provider encrypts s_i under each of the $4k$ keys in row A_{r_i}.

Decryption: Each authorized user has one key from every row A_{r_i} and is therefore always able to decrypt every s_i and compute s.

Parameters: The memory required per user is $m = \ell$ keys. The amount of work that each user should perform in order to reveal a key is $O(t)$. The data redundancy overhead used in distributing the key is $r = 4kt$.

The parameter t should be set so that for t random rows it holds with probability q that a pirate decoder which contains keys from less than a fraction w of the rows does not have a key from at least one of the t rows (and therefore a decoder which does not have keys from a fraction w of the rows cannot decrypt with probability better than q). First observe that $w \geq q$ since otherwise the probability is less than q even for $t = 1$. The probability of the decoder having keys from all t rows is at most w^t and therefore setting $t = \log_w q = \frac{\log(1/q)}{\log(1/w)}$ suffices to make the probability of correct decryption at most q. For example, it is possible to set $w = q$ and $t = 1$. The broadcast center would only have to broadcast the secret s encrypted by the keys of a single row which it chooses randomly. The data redundancy overhead is then only $O(4k)$.

Tracing: We are only concerned with decoders which include keys from at least $w\ell$ rows[4]. Using the methods of [9] it is possible to reveal the set of keys F that a pirate decoder uses while treating the decoder as a black box. Assume w.l.o.g. that F contains one key from each of $w\ell$ rows. Denote these rows as $r_1, \ldots, r_{w\ell}$, and denote the key in $F \cap A_{r_i}$ as f_{r_i}. The body that performs the traitor tracing knows the functions $h_{r_i}(\cdot)$ and can therefore identify and mark the users in $h_{r_i}^{-1}(f_{r_i})$ for every i. The user with the largest number of marks is exposed as the traitor.

Analysis: Since there were at most k traitors it is obvious that one of them contributed $w\ell/k$ keys to F. Consider the probability that an innocent user, say user 1, contributed $w\ell/k$ keys to F. Since the hash functions h_{r_i} are random and secret the mapping $h_{r_i}(1)$ is random and independent of the mapping of the traitors by h_{r_i}. The probability that f_{r_i} equals the key mapped to user 1 is $1/4k$. An immediate application of the Chernoff bound shows that the probability that at least $w\ell/k$ of the keys of user 1 are in F is at most $2^{-3w\ell/4k}$. Choosing an ℓ such that $n \cdot 2^{-3w\ell/4k} < p$ ensures a traitor is revealed with probability at least $1 - p$. The data provider should therefore set $\ell = \frac{4k}{3w}\log(n/p)$.

For any practical purpose the parameter q can be set to be a constant. However one-level schemes are used in the next subsection as building blocks for two-level schemes and there q should be a function of other parameters. The results regarding one-level threshold schemes are summed up in the following theorem. We first state the results for a parameterized w. As w increases the key length decreases and the data redundancy overhead increases. Then we state the results for $w = q$.

Theorem 2. *There is a q-threshold (p, k)-resilient scheme, with a parameter w taking values in $[q, 1)$, in which a personal key consists of $\frac{4k}{3w}\log(n/p)$ keys and the data redundancy overhead is of $4k\frac{\log(1/q)}{\log(1/w)}$ keys. A user should perform $\frac{\log(1/q)}{\log(1/w)}$ decryptions in order to reveal the broadcasted secret.*

When $w = q$ a personal key consists of $\frac{4k}{3q}\log(n/p)$ keys and the data redundancy overhead is of only $4k$ keys. A user should only perform a single decryption in order to decrypt the broadcasted secret.

The scheme we presented displays a tremendous improvement in the data redundancy overhead, but the length of the personal key is a little larger than in the one-level fully resilient scheme (it is $\frac{4k}{3q}\log(n/p)$ compared to $\frac{4k}{3}\log(n/p)$ in the

[4] It is possible to prove, as is done in [9], that if a decoder has keys from less rows and can decrypt with probability better than the threshold then it can be used to break the underlying encryption scheme.

one-level fully resilient scheme). The next subsection presents two-level threshold schemes which balance the two complexity parameters through a tradeoff between the key length and the data redundancy overhead.

4.2 Two-Level Threshold Schemes

Two-level threshold schemes are constructed from one-level threshold schemes by using many one-level schemes and applying a hash function to map users to schemes. We first present a basic construction which displays a tradeoff between the personal key lengths and the data redundancy overhead, and which can obtain shorter key length than the one-level threshold scheme. Then we change the parameters of the construction to obtain schemes with an even shorter key length, in the price of increasing the data redundancy a little. These schemes perform better than fully-resilient schemes in both the personal key length and the data redundancy overhead.

The basic construction The construction uses a random mapping h from the domain $\{1,\ldots,n\}$ to a range of size $2ek/b$. It is required that for any fixed set of k traitors the probability that b or more traitors are mapped together by h is less than $p/2$, i.e.

$$\binom{k}{b}\left(\frac{b}{2ek}\right)^{b-1} < \left(\frac{ek}{b}\right)^{b}\left(\frac{b}{2ek}\right)^{b-1} = \frac{ek}{b}\frac{1}{2^{b-1}} < \frac{p}{2}$$

Setting $b = \log\left(\frac{4ek}{p\log(1/p)}\right)$ satisfies the inequality. Once such a mapping is chosen we continue by constructing threshold one-level schemes for each set of preimages $h^{-1}(i)$ for $1 \leq i \leq 2ek/b$. In the initialization phase each user u receives his personal key for the subscheme $h(u)$, and the secret s is distributed by *each* of the $2ek/b$ subschemes.

It is required that each subscheme has the following property against b traitors: either the success probability of the traitors in decrypting the secret is greater by at most $\tilde{q} = \frac{qb}{2ek}$ than the success probability of an adversary who does not have any of the keys, or the traitors can be traced with probability at least $1 - p/2$. If in no subscheme the traitors have an advantage greater than $\tilde{q}$ then the pirate decoder cannot decrypt with an advantage better than q.

The initialization and secret distribution stages are straightforward. The subschemes are built in the same way as the one-level schemes of the previous subsection. As before w is a parameter that defines the minimal fraction of rows such that with keys from less than $w\ell$ rows in a certain subscheme a decoder cannot decrypt with probability better than $\tilde{q}$. If a pirate decoder decrypts with probability greater than q it must contain keys from a w fraction of the rows in one or more of the subschemes. The tracing process that was defined for the one-level scheme can then trace at least one of the traitors which contributed keys for this subscheme. The following theorem therefore follows:

Theorem 3. *There is a q-threshold (p,k)-resilient scheme, with the parameter w taking values in $[\frac{qb}{2ek},1)$, where $b=\log(\frac{4ek}{p\log(1/p)})$, in which:*

- *The length of the personal key is $m=\frac{4}{3w}b\log(2n/p)$ basic keys.*
- *The data redundancy overhead is $8ek\frac{1}{\log(1/w)}\log(\frac{2ek}{qb})$ basic encryptions.*
- *The receiver should perform $\frac{\log(2ek/(qb))}{\log(1/w)}$ decryptions in order to decrypt the secret.*

The key is longer than the key in the fully resilient secret two-level scheme by a factor of only $1/w$, and the data redundancy overhead is substantially shorter. Comparing with the one-level threshold scheme for the same value of the parameter w, the personal key changes by a factor of b/k, and the data redundancy overhead changes by a factor of $2e\cdot(1+\log(2ek/b)/\log(1/q))$. Therefore the key is shorter and the data redundancy overhead is larger. However, the increase in the data redundancy overhead is relatively moderate: if we denote the ratio between the key length in this scheme and in the one-level scheme as $1/\alpha$ then the data redundancy overhead increases by a factor of only $2e(1+\log(2e\alpha)/\log(1/q))$.

Note that the minimum value for w is $\tilde{q}=\frac{qb}{2ek}$ which is smaller than the minimum value for w in the one-level scheme. Setting w to this value yields the minimum possible data redundancy overhead, $8ek$ encryptions, whereas the key length is maximal, $m=\frac{8ek}{3q}\log(2n/p)$. Both are longer than the values for the one-level scheme by a factor of exactly $2e$.

The two-level scheme features a tradeoff between the length of the personal key and the data redundancy overhead. At one end of the curve there is a short key but a longer data redundancy overhead and in the other end the key length is maximal and the data redundancy overhead is minimal, and both are equal up to a constant factor to the performance of the one-level threshold scheme for minimal data redundancy overhead. Note that as with the two-level fully-resilient secret scheme the expected number of users that are mapped to each subscheme is smaller than n by a factor of $b/2ek$. The subschemes can therefore be defined for a smaller set of users to achieve greater efficiency.

Shorter personal keys The following variant of a threshold tracing scheme improves all the complexity parameters of the most efficient fully-resilient scheme (whereas the previous tracing scheme had a dramatic improvement in the data redundancy and decryption overheads, but increased the personal key a little). The decrease in the length of the personal keys is enabled as follows: The same construction as before is used, with $2ek/b_1$ subschemes, and it is required that the probability that more than b_2 users are mapped together is at most $p/2$ (previously the values b_1 and b_2 were equal). The personal key is composed of $\frac{4}{3w}b_2\log(2n/p)$ keys, and the data redundancy overhead is of $8ek\frac{b_2}{b_1}\frac{1}{\log(1/w)}\log(\frac{2ek}{qb_1})$ basic encryptions.

The values b_1,b_2 should satisfy the following inequality:

$$\binom{k}{b_2}\cdot\left(\frac{b_1}{2ek}\right)^{b_2-1}\le\left(\frac{ek}{b_2}\right)^{b_2}\cdot\left(\frac{b_1}{2ek}\right)^{b_2-1}=\frac{2ek}{b_1}\cdot\left(\frac{b_1}{2b_2}\right)^{b_2}<\frac{p}{2}$$

Assume $b_2 = b_1^\alpha = b^\alpha$ $(\alpha > 1)$. The previous inequality is satisfied if $b \geq \sqrt[\alpha]{\frac{\alpha}{\alpha-1} \cdot \frac{\log(k/p)}{\log\log(k/p)}}$. The following theorem is therefore obtained:

Theorem 4. *For every $\alpha > 1$ there is a q-threshold (p,k)-resilient scheme, with the parameter w taking values in $[\frac{qb}{2ek}, 1)$, where $b = \sqrt[\alpha]{\frac{\alpha}{\alpha-1} \cdot \frac{\log(k/p)}{\log\log(k/p)}}$, in which:*

- *The length of the personal key is $m = \frac{4}{3w} \cdot b^\alpha \cdot \log(2n/p)$ basic keys.*
- *The data redundancy overhead is $8ekb^{\alpha-1}\log(\frac{2ek}{qb})/\log(1/w)$ basic encryptions.*
- *A receiver should perform $\log(\frac{2ek}{qb})/\log(1/w)$ decryptions in order to decrypt the secret.*

As α increases the personal key length decreases and the data redundancy overhead increases. The limits of these values as $\alpha \longrightarrow \infty$ are

- *The limit of the length of the personal key is $m = \frac{4}{3w} \cdot \frac{\log(k/p)}{\log\log(k/p)} \cdot \log(2n/p)$ basic keys.*
- *The limit of the data redundancy overhead is $8ek\frac{\log(k/p)}{\log\log(k/p)}\log(\frac{2ek}{q}) \cdot \frac{1}{\log(1/w)}$ basic encryptions.*
- *A receiver should perform $\frac{\log(2ek/q)}{\log(1/w)}$ decryptions in order to decrypt the secret.*

This scheme has the shortest personal key among all the schemes we presented. The small penalty for this is a data redundancy overhead which is longer than in the other threshold two-level scheme. However, the data redundancy is still shorter than in the fully resilient schemes.

4.3 An Example

Let us consider the following example in order to demonstrate the performance of the different tracing schemes. Suppose that we would like to create a traitor tracing scheme for up to one million authorized users, so that for at most $k = 1000$ traitors the probability of false identification is at most 2^{-10}. We describe in Table 1 the length of the personal key of each user and the data redundancy overhead, both measured by the number of basic keys that they contain (i.e. the ratio between their size and the size of a key of the encryption scheme that is used to encrypt the content). The table also shows the number of decryption operations that should be performed by the receiver. We compare the performance of threshold schemes to the performance of the best fully-resilient scheme – the two-level secret scheme described in section 3. The table refers to the section in which each of the schemes is described. The first result is for the most efficient two-level secret fully resilient scheme. The other results are of threshold schemes which were designed to trace only the source of keys of decoders which can decrypt with probability greater than 3/4. This type of schemes allows for

	Property	Section	Personal Key	Data Redun.	Decryption Operations
Secret two-level	best fully-res.	3	496	21,270,000	496
Threshold	one-level, min. data redundancy	4.1	53,000	4000	1
Threshold	two-level, min. data redundancy	4.2 $w = 1/2$	1,660	185,000	9
Threshold	two-level min. key	4.2 $\alpha \to \infty$	380	1,290,000	13
Threshold	tradeoff	4.2 $w = 1/8$	10,000	64,500	3

Table 1. Examples of the complexity of different Tracing Traitors schemes, using $n = 10^6, k = 1000, p = 10^{-3}$, and $q = 3/4$.

a tradeoff between the length of the personal key and the data redundancy, as is demonstrated in the table.

The secret two-level scheme has a short key length but the data redundancy overhead is large. The threshold schemes feature a tradeoff between the length of the personal key and the data redundancy overhead. It is possible to make one parameter very small by increasing the other parameter, and it is also possible to achieve very reasonable results for both measures, as in the last entry. The scheme of Section 4.2 is superior to the secret two-level scheme in all the complexity parameters. It should also be noted that if we are only concerned with decoders which decrypt with probability close to 1 it is possible to get more efficient schemes by defining a scheme for $q \approx 1$.

5 Conclusions

We presented threshold tracing schemes which are considerably more efficient than fully-resilient tracing schemes. In many applications there is only need for decoders which decrypt with probability greater than some threshold, and these applications should use threshold tracing schemes to trace the source of illegal decoders. The efficiency of the threshold schemes as a function the size of a corrupt coalition of users, k, allows for resiliency against rather large such coalitions.

We remark that in many different applications and scenarios (other than traitor tracing) there is no need for security against adversaries which perform negligibly better than "guessing the secret". These applications call for threshold security schemes similar to the schemes presented in this work. These schemes should depend on a parameter q (the threshold) and only protect against adversaries which achieve success greater than q.

References

1. N. Alon, J. Bruck, J. Naor, M. Naor and R. Roth, *Construction of Asymptotically Good Low-Rate Error-Correcting Codes through Pseudo-Random Graphs*, IEEE Transactions on Information Theory, vol. 38 (1992), 509–516.
2. N. Alon and J. Spencer, **The Probabilistic Method**, Wiley, 1992.
3. R. Anderson and M. Kuhn, *Tamper Resistance – A Cautionary Note*, Usenix Electronic Commerce Workshop, Oakland (1996), 1–11.
4. E. Biham and A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, Proc. Advances in Cryptology – Crypto '97, Springr-Verlag LNCS 1294 (1997), 513–525.
5. D. Boneh, R. A. Demillo and R. J. Lipton, *On the Importance of Checking Computations*, Proc. Advances in Cryptology – Eurocrypt '97 (1997), 37–51.
6. D. Boneh and J. Shaw, *Collusion-Secure Fingerprinting for Digital date*, Proc. Advances in Cryptology – Crypto '95 (1995), 452–465.
7. J. L. Carter and M. N. Wegman, *Universal Classes of Hash Functions*, Journal of Computer and System Sciences 18 (1979), 143–154.
8. B. Chor, A. Fiat and M. Naor, *Tracing Traitors*, Proc. Advances in Cryptology – Crypto '94, Springr-Verlag LNCS 839 (1994), 257–270.
9. B. Chor, A. Fiat, M. Naor and B. Pinkas, *Tracing Traitors*, manuscript, (1998).
10. Cox I., Kilian J., Leighton T. and Shamoon T., *A Secure, Robust Watermark for Multimedia*, Information Hiding Workshop, Cambridge, UK, Springer-Verlag LNCS 1174, (1996), 185–206.
11. C. Dwork, J. Lotspiech and M. Naor, *Digital Signets: Self-Enforcing Protection of Digital Information*, 28th Symposium on the Theory of Computation (1996), 489–498.
12. P. Erdös, P. Frankl and Z. Füredi, *Families of finite sets in which no set is covered by the union of r others*, Israel J. of math. **51** (1985), 79–89.
13. A. Fiat and M. Naor, *Broadcast Encryption*, Proc. Advances in Cryptology - Crypto '93 (1994), 480–491.
14. M.L. Fredman, J. Komlós and E. Szemerédi, *Storing a Sparse Table with $O(1)$ Worst Case Access Time*, Journal of the ACM, Vol 31 (1984), 538–544.
15. K. Mehlhorn, **Data Structures and Algorithms: Sorting and Searching**, Springer-Verlag (1984).
16. F. J. MacWilliams and N. J. A. Sloane, **The Theory of Error Correcting Codes**, North Holland, Amsterdam, (1977).
17. B. Pfitzmann, *Trials of Traced Traitors*, Information Hiding Workshop, Cambridge, UK, Springer-Verlag LNCS 1174, (1996), 49–64.
18. D.M. Wallner, E.J. Harder, R.C. Agee, *Key Management for Multicast: Issues and Architectures*, internet draft `draft-wallner-key-arch-00.txt` (1997). Avaliable at `ftp://ietf.org/internet-drafts/draft-wallner-key-arch-00.txt`
19. M. N. Wegman and J. L. Carter, *New Hash Functions and Their Use in Authentication and Set Equality*, Journal of Computer and System Sciences 22 (1981), 265–279.