Tracing Traitors

Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas

***Abstract*—We give cryptographic schemes that help trace the source of leaks when sensitive or proprietary data is made available to a large set of parties. A very relevant application is in the context of pay television, where only paying customers should be able to view certain programs. In this application, the programs are normally encrypted, and then the sensitive data is the decryption keys that are given to paying customers. If a pirate decoder is found, it is desirable to reveal the source of its decryption keys.**

We describe *fully resilient* schemes which can be used against any decoder which decrypts with nonnegligible probability. Since there is typically little demand for decoders which decrypt only a small fraction of the transmissions (even if it is nonnegligible), we further introduce *threshold* tracing schemes which can only be used against decoders which succeed in decryption with probability greater than some threshold. Threshold schemes are considerably more efficient than fully resilient schemes.

***Index Terms*—Encryption, tracing, watermarking.**

I. Introduction

If only one person knows some secret, and this next appears on the evening news, then the guilty party is evident. A more complex situation arises if the set of people that have access to the secret is large. The problem of determining guilt or innocence is (mathematically) insurmountable if all people get the exact same data and one of them behaves treacherously and reveals the secret.

Any data that is to be available to some while it should not be available to others can obviously be protected by encryption. The *data supplier* may give authorized parties cryptographic keys allowing them to decrypt the data. This does not solve the problem above because it does not prevent one of those authorized to view the message (say, Alice) from transferring the *cleartext* message to some unauthorized party (say, Bob). Once this is done then there are no (cryptographic) means to trace the source of the leak. We call all such unauthorized access to data *piracy*. The *traitor* or *traitors* is the (set of) authorized user(s) who allow other, nonauthorized parties, to obtain the data. These nonauthorized parties are called *pirate users*.

In many interesting cases piracy is somewhat ineffective if the relevant *cleartext* messages must be transmitted by the "traitor" to the "enemy." Typical cases where this is true include

- Pay-per-view or subscription television broadcasts. It is simply too expensive and risky to start a pirate broadcast station.
- Online databases, publicly accessible (say on the Internet) where a charge may be levied for access to all or certain records.
- Distribution of data in an encrypted form where a surcharge is charged for the decryption keys for different parts of the data. The encrypted data is often distributed on a CD-ROM or DVD and it is assumed that cleartext data can only be distributed on a similar storage device whose production involves relatively high setup costs (this assumption might not be currently justified for CD-ROM's but it might be reasonable for other types of media, such as DVD's. We use the term CD-ROM in order to use a concrete example and simplify the presentation).

In all these cases, transmitting the cleartext from a traitor, Alice, to a pirate-user, Bob, is rather expensive compared to the mass distribution channels the legal data supplier uses. It might also be the case, as with on-line databases or newspapers, that the data is continuously changing and therefore it is very hard for the pirate to keep an updated copy of the data. As piracy in all these cases is a criminal commercial enterprise, the risk/benefit ratio becomes very unattractive. These three examples can be considered generic examples covering a wide range of data services.

In this paper we concentrate on preventing traitors from distributing the *keys* that enable the decryption of the encrypted content. Consider a ciphertext that may be decrypted by a large set of parties, but each and every party is assigned a different *personal key* for decrypting the ciphertext. (We use the term personal key rather than private key to avoid confusion with public key terminology). Should the key used in a pirate decoder be discovered (by examining the pirate decoder or by counter-espionage), it will be linked to a personal key of a traitor and this traitor will be identified.

Clearly, a possible solution is to encrypt the data separately under different personal keys. This means that the total length of the ciphertext is at least n times the length of the cleartext, where n is the number of authorized parties. Such overhead is certainly impossible in any broadcast environment. It is also very problematic in the context of content distributed on a DVD because this means that every copy must be different. An encrypted online database, publicly accessible as above, must store an indi-

Manuscript received March 29, 1998; revised November 28, 1999. The work of B. Chor was supported by the Fund for Promotion of Research at the Technion. The work of M. Naor was supported under a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences and by an Alon Fellowship. The work of B. Pinkas was supported by an Eshkol Fellowship from the Israeli Ministry of Science. The material in this paper was presnted in part at Crypto'94 and Crypto'98.

B. Chor is with the Department of Computer Science, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: benny@cs.technion.ac.il).

A. Fiat is with the Department of Computer Science, School of Mathematics, Tel-Aviv University, Ramat-Aviv 69978, Israel, and Algorithmic Research Ltd. (e-mail: fiat@math.tau.ac.il).

M. Naor and B. Pinkas are with the Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel (e-mail: {naor}{bennyp}@wisdom.weizmann.ac.il).

Communicated by D. Stinson, Associate Editor for Complexity and Cryptography.

Publisher Item Identifier S 0018-9448(00)03089-3.

vidually encrypted copy of the database for each and every authorized user.

In practice today it is often considered sufficient to prevent piracy by supplying the authorized parties with so-called secure hardware solutions that are designed to prevent interference and access to enclosed cryptographic keys (smartcards and their like). 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], [18]. 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.

1) Our Approach: Fighting piracy in general has the following components.

a) Identifying that piracy is going on and preventing the transmittal of information to pirate users, while harming no legitimate users.

b) Taking measures against the source of such piracy, supplying legal evidence of the pirate identity.

The goal of this paper is to deal with *traitor tracing* (item a) above), i.e., identify the source of the problem. Methods that can be taken in order to eliminate pirate decryption of the content are described in Section I-A.

We devise *k-resilient traceability* schemes which, loosely speaking, have the following properties.

- Either the cleartext information itself is transmitted to the pirate users by a traitor, or
- Any captured pirate decoder (which decrypts with success probability which is better than the probability of breaking the encryption scheme that is used) will correctly identify a traitor and will protect the innocent even if up to k traitors collude and combine their keys.

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. In order to identify a traitor, it suffices to capture one pirate decoder and examine its behavior; there is no need to "break it open" or read any data stored inside. (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).

The underlying security assumption of our schemes is either information-theoretic security (where the length of the personal keys grows with the length of the messages to be transmitted) or it may be based on the security of any symmetric encryption scheme. In both cases, security depends on a parameter k, denoting the largest group of colliding traitors.

The security of the scheme depends on a cryptographic security parameter which is the length of the key in the symmetric encryption system that is used. We measure the *efficiency* of the solutions to fighting piracy 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 as follows.

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 offline 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.

We deal with schemes 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 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 scheme consists of pairs of the form (*enabling block, cipher block*). The cipher block is the symmetric encryption of the actual data (say a few seconds 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. The computation on the user end, for all schemes we present, is simply taking the exclusive–or of values that the user is able to decrypt.

Fig. 1 describes a high-level view of our traitor tracing schemes. Fig. 2 describes a high-level view of a single decoding box.

Traitors may conspire and give an unauthorized user (or users) a subset of their keys, so that the unauthorized user will also be ables to compute the real message key 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 should be possible to detect at least one traitor, subject to the limitation that the number of traitors is, at most, k. (We cannot hope to detect all traitors as one traitor may simply provide his personal key and others may provide nothing).

We remark that in many cases it is preferable to predetermine a fixed number of users n, and to assign them personal keys, even if the actual number of users is smaller. Users who join the system later (e.g., by purchasing a subscription to a television station or an online database) are assigned personal keys from those pre-installed.

2) Threshold Tracing: We further distinguish between two kinds of tracing schemes. *Fully resilient* schemes guarantee the tracing of the source of any pirate decoder which decrypts with nonnegligible success probability (more accurately, which per-

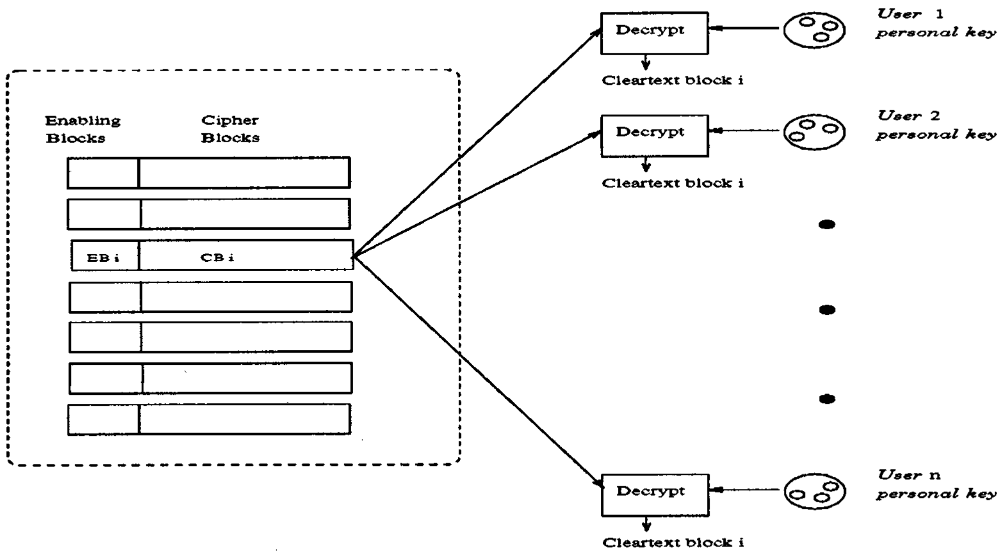

Fig. 1. A high-level view of the traitor tracing scheme.

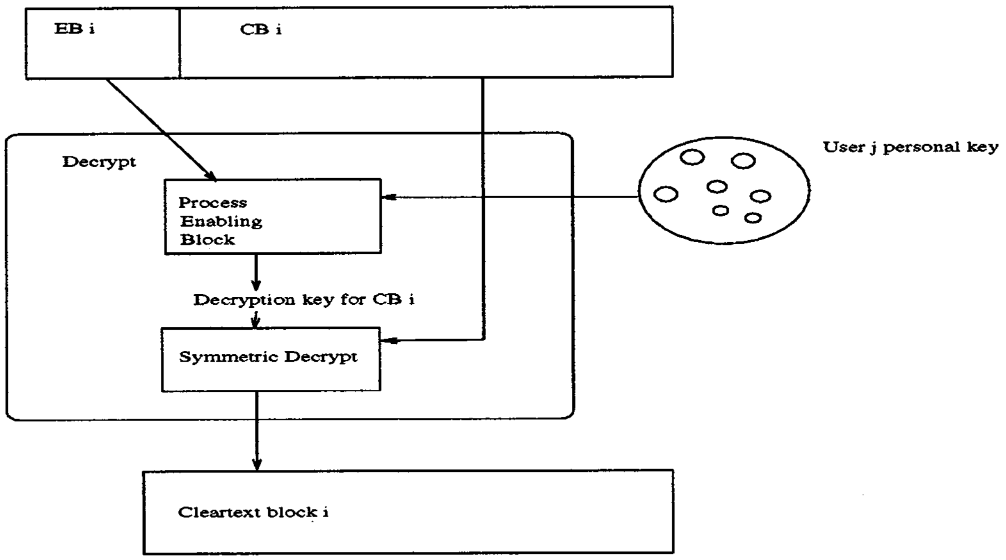

Fig. 2. The operation of a single decoding box.

forms better than breaking the underlying encryption system that is used to encrypt the data). However, in many applications such security is not needed and it is enough to fight pirate decoders which have a considerable success probability. For example, in pay-TV applications pirate decoders which decrypt only part of the content are probably useless. We therefore also demonstrate *threshold* schemes which only trace the source of the keys of decoders which decrypt with probability greater than some threshold q, which is a parameter of the scheme. These schemes are considerably more efficient than fully resilient schemes.

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.

A demonstration of the efficiency of threshold tracing schemes compared to that of the best fully resilient tracing scheme appears in Table I. This table presents the exact overhead of the schemes, for a system of typical size. A comparison of the asymptotic behavior of the overheads of all our schemes appears in Table II.

A. 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.

TABLE I
EXAMPLES OF THE COMPLEXITY OF DIFFERENT TRACING TRAITORS SCHEMES, USING $n = 10^6, k = 500, p = 10^{-3}, q = 3/4$

	PROPERTY	SECTION	PERSONAL KEY	DATA REDUN.	DECRYPTION OPERATIONS
Trivial			1	1,000,000	1
Secret two-level	best fully-res.	V-B	493	11,300,000	493
Threshold	one-level, min. data redun.	VI-A	26,500	2000	1
Threshold	two-level min. key	VI-B1 $w = 1/2$	1,570	82,000	8
Threshold	two-level min. key	VI-B2 $\alpha \to \infty$	370	574,000	12
Threshold	tradeoff	VI-B1 $w = 1/8$	6,300	27,500	3

TABLE II
A COMPARISON OF THE DIFFERENT TRACING SCHEMES. THE PARAMETERS INCLUDE n—THE NUMBER OF COPIES, k—THE MAXIMUM NUMBER OF COPIES KNOWN TO PIRATES, p—THE PROBABILITY THAT PIRATES CANNOT BE TRACED. THE THRESHOLD SCHEMES ARE DESIGNED FOR A CONSTANT THRESHOLD

SCHEME	KEY	DATA REDUNDANCY	DECRYPTION
trivial	1	n	1
open one-level	$O(k^2 \log n)$	$O(k^4 \log n)$	$O(k^2 \log n)$
open two-level	$O(k^2 \log^2 k \log(n/k))$	$O(k^3 \log^4 k \log(n/k))$	$O(k^2 \log^2 k \log(n/k))$
secret one-level	$O(k \log(n/p))$	$O(k^2 \log(n/p))$	$O(k \log(n/p))$
secret two-level	$O(\log(n/p) \log(1/p))$	$O(k \log(n/p) \log(1/p))$	$O(\log(n/p) \log(1/p))$
threshold one-level	$O(k \log(n/p))$	$O(k)$	$O(1)$
threshold two-level	$O(\log(\frac{k}{p \log(1/p)}) \log(n/p))$	$O(k \log(k/ \log(\frac{k}{p \log(1/p)})))$	$O(\log(k/ \log(\frac{k}{p \log(1/p)})))$

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, a 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 the source of its keys is identified, legal action should be taken against this source. Indeed, as was pointed by Pfitzmann in [22], 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. Note, however, that similar misbehavior is possible with many (maybe even all) current types of services, and yet there is little evidence that service providers have performed such illegal activities. Even if this possibility would weaken the legal status of evidences found using tracing traitors schemes, the data provider itself can use this evidence to identify the pirates and then try to obtain other types of legal proofs about their activities. There has also been some work in suggesting tracing traitors schemes which do not enable the data provider to fabricate evidence against honest users, but they are still far from being applicable. See Section I-B for details.

If a pirate user has obtained content in encrypted form and all the keys that are required to decrypt it, there is little one can technically do to prevent her from continuing to use the content. In this case, the only remedy is legal. The situation is somewhat different if the system requires some action on behalf of the data supplier, as with a television broadcast or online database. We call such cases "active data." Such systems might allow to disable identified pirate users from further receiving content.

The broadcast encryption schemes of Fiat and Naor [14] 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.* [25] (and improved in [8]), and is secure against any number of corrupt users. When used for n receivers it requires each receiver to store only $\log n$ keys. There is no data redundancy in regular transmission of data, and whenever a user should be deleted from the set of legitimate receivers, the scheme sends a single message of length $O(\log n)$ which generates a new key which is unknown to the deleted user. The communication overhead is therefore $O(\log n)$ times the number of users which are removed from the set of receivers. Since we assume that there would be a modest number of traitors (at most k), this scheme is well suited to efficiently handle their deletion from the privileged set of receivers (whereas the scheme of [14] has better performance for deleting a large group of receivers).

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 dropping the traced traitor from it. This procedure should be repeated until the pirate box is rendered useless. Since no innocent user is labeled as a traitor (at least with high probability), the operation of legitimate users is not interrupted. Such a combination, however, cannot be constructed by simply taking each session key as the bit-wise exclusive–or of keys transmitted by the traitor tracing scheme and the broadcast encryption scheme. The drawback of such a simple solution is that a pirate can use different sets of keys in the parts of the decryption box that decrypt the tracing traitors and the broadcast encryption schemes. The data provider can only identify the keys that are used for the tracing traitors scheme, but cannot render them useless since this operation is only possible for keys used in the broadcast encryption scheme.

It is possible to combine the tracing traitors and the broadcast encryption schemes in a different way: The tracing schemes we describe operate by distributing the secret into many shares and encrypting each share with several keys. Every legitimate user u receives a set of keys $P(u)$ which enable them to decrypt enough shares to reveal the secret, and the scheme ensures that the keys of a pirate decoder identify at least one of the traitors who contributed these keys. In order to combine broadcast encryption, each share should be encrypted by a different broadcast encryption scheme. For every key which was previously in $P(u)$, the combined scheme should provide the user u with decryption keys for the corresponding broadcast encryption scheme. When a traitor t is revealed, it should be deleted from the sets of receivers in the broadcast encryption schemes corresponding to $P(t)$. The length of a personal key of a user is, therefore, the product of the lengths of the personal keys in the tracing traitors and broadcast encryption schemes. Similarly, the data redundancy overhead is the product of the data redundancy overheads in the two schemes. Note that the broadcast encryption schemes of [25], [8] require a personal key of length $\log n$, and during normal operation the data redundancy is $O(1)$. Therefore, combining this scheme with our schemes requires each user to store $\log n$ as many keys, does not increase the data redundancy during normal operation, and requires a special message of length $O(\log n)$ times the size of the personal key whenever a traitor is revealed and deleted from the system.

B. Related Work

The current work appeared in a preliminary form in [10] and in [21]. Some related work followed the initial publication of our traitor tracing schemes in [10].

Boneh and Shaw [7] have suggested a scheme for fingerprinting different copies of an electronic document by inserting a different watermark into each copy. Their scheme has the property that using up to k copies it is impossible (with some probability) to generate a new copy whose fingerprint does not reveal at least one of the k copies that were used. The scheme offers better security in the sense that it makes it possible to trace the leaked content itself (and not just the key which enables its decryption). However, it is based on a marking assumption.[1] It can also be used as a tracing traitors scheme, but it is much less efficient than our schemes, the number of keys that each user should have is k^4 times greater than in our two-level secret scheme.

Another solution for copyright protection is through *self-enforcement* schemes, which were suggested by Dwork, Lotspiech, and Naor [12]. 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 [12] use the same type of security as we use in this paper. Namely, the system is secure against coalitions of less than k corrupt users, and the system's complexity depends on k. The signets scheme of [12] is less efficient than our tracing schemes. The lengths of the personal key and of the data redundancy in the signets scheme are k times the total size of secrets which are sent using the scheme.

Pfitzmann [22] 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 that the decoder contains. However,

[1]See, for instance, [11] for a method of inserting marks into a document.

as this construction uses a relatively complex primitive (general secure multiparty protocols) which is rather inefficient (e.g., it operates on the circuit which evaluates the function), its overall complexity is high.

The combinatorial properties of tracing schemes were investigated by Stinson and Wei in [23], and Staddon [24] investigated the relations between combinatorial tracing schemes and broadcast encryption schemes. Boneh and Franklin [6] presented public key tracing schemes, which enable public key encryption and, being based on a number-theoretic assumption, are more efficient than combinatorial tracing schemes (including those presented in this paper). Finally, Fiat and Tassa [15] introduced dynamic tracing schemes in which, in order to locate the traitor, the tracing algorithm dynamically changes the content that is being broadcast to different subsets of the users. These schemes enable tracing even if the traitor is revealing the content itself and not only the keys that encrypt it.

C. 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 $k = 500$ traitors, the probability of false identification is, at most, 2^{-10}. Table I describes the length of the personal key of each user and the data redundancy overhead, both measured by the number of basic keys they contain, and also the number of decryption operations that are performed by the receiver. Since we describe both fully resilient and threshold tracing schemes, we compare the performance of threshold schemes to the performance of the best fully resilient scheme—the two-level secret scheme described in Section V-B. The table refers to the section in which each of the schemes is described. The first row describes the overhead of the trivial solution of independently encrypting the secret to every user. The second row describes the most efficient two-level, fully resilient, secret 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 a tradeoff between the length of the personal key and the data redundancy, as is demonstrated in the table.

The fully resilient scheme has a short key length, but the data redundancy overhead is quite large. In fact, for the example described in Table I, the data redundancy is larger than in the trivial scheme in which each user has a different independent key. However, this scheme is not too bad if it is used for a high-bandwidth channel, and parameters for which it performs better than the trivial scheme (namely, smaller values of k). 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 VI–B1 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 closer to 1, it is possible to obtain even more efficient schemes.

II. 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 $P_\alpha : U \mapsto \{0,1\}^s$ where U is the set of possible users and s is the number of bits in the personal key that each users gets. When user $u_i \in U$ joins, she receives her 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, combining 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 it can access it as a black box and test how it decrypts an input ciphertext. (We do assume, however, that the pirate decoder can be reset to its original state, i.e., we assume that there is no self-destruction mechanism which is triggered when it detects a traitor tracing algorithm.)

The encryption of plaintext blocks 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 keyed by a random *block key*, 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.

Fully resilient tracing: Assume that an adversary has the cooperation of a coalition of at most k users, and uses their keys to construct a decoder. We would like to be able to trace at least one of the coalition members. Intuitively, we call a scheme *fully resilient* if we can 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, we say that a system is fully resilient if for every pirate decoder which runs in time t it either holds that we can trace at least one of the traitors which helped to build it, or that the decoder can break one of the underlying encryption algorithms in time t.

Threshold tracing: There are many applications in which the pirate decoder must decrypt with probability close to 1. For example, if a TV broadcast is partitioned into short segments and these segments are encrypted independently, then customers would not buy a decoder which decrypts only 90% of the segments. In such scenarios, we can concentrate on decoders which can 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, we can trace at least one of the traitors that helped build it.

An obvious and preliminary requirement from 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 being traced, then the scheme is insecure as an encryption scheme, *even against an adversary who has no keys*.

Definition 1 (Fully (p, k)-Resilient Tracing Scheme): Let T be a coalition of at most k users. Let $\mathcal{A}$ be an adversary that has a subset F of the keys of the users in T, and that is able to decrypt the content sent in the tracing traitors scheme, in time $\boldsymbol{t}$ and with probability greater than $\boldsymbol{q'}$.

The scheme is called *fully (p, k)-resilient* if it satisfies the following security assumption.

The Security assumption: *one of the following two statements holds.*

- Given F the data supplier is able to trace with probability at least $\boldsymbol{1 - p}$ 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 $\boldsymbol{t}$, and with probability at least $\boldsymbol{q'' = q'}$.

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.

Definition 2 (Fully k-Resilient Tracing Scheme): A scheme is called *fully k-resilient* if it satisfies Definition 1, and it further holds that $p = 0$.

Definition 3 (q -Threshold (p, k)-Resilient Tracing Scheme): A scheme is called *q -threshold (p, k)-resilient* if it satisfies Definition 1 with $q'' = q' - q$.

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 of the content) 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 nonnegligible probability. In a threshold scheme the data supplier is able to do so if it finds a decoder which decrypts with probability which is greater than q by a nonnegligible difference.

We further distinguish between two types of schemes: the first type, called an *open scheme*, treats circumstances where the decryption schemes used by all users are in the public domain, and the decryption keys themselves are the only information that is kept secret. The second type is where the actual decryption scheme as well as the keys are kept secret, and it is called a *secret scheme*. In particular, in open schemes it is publicly known which keys (from the base set of keys A) are contained in each decoder, whereas in secret schemes this information is kept secret.

Since the goal of an adversary is to prevent the traitors from being identified, one way to ensure this is to incriminate someone else. Clearly, the adversary's task is no harder with an open scheme compared to a secret scheme. On the other hand, secret schemes pose additional security requirements at the data supplier site.

We present efficient fully resilient schemes of both types. The open schemes are fully k-resilient (that is, they always trace at least one of the traitors). However, our constructions of secret schemes are much more efficient. We also present q-threshold schemes which fall into the category of secret schemes, and these schemes have even better performance. It is clearly advantageous to use secret schemes in practice, and any real implementation will do so. In addition, whenever the application enables us to use threshold schemes, it is preferable to use them since they provide even better performance.

III. Overview of the Results

Throughout the paper, we denote by k an upper bound on the number of traitors. Every enabling block consists of r encryptions, and m denotes the number of keys comprising a user's personal key.

We describe six k-resilient traceability schemes. A concise listing of the personal key length and the data redundancy overhead required by each scheme appears in Table II. All the schemes are based on the usage of hash functions combined with *any* private key cryptosystem, and do not require the use of public key operations. For more information on hash functions and their applications, see [19], [9], [26], and [16]. The basic usage of hash functions is to assign 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 (nontraitor) user.

The first four schemes are fully resilient and trace the sources of the keys of any pirate decoder which is able to decrypt with nonnegligible probability. Note that in these scheme the length of the personal key stored by the user is the same as the number of operations that a user should perform in order to reveal the transmitted secret. The last two are threshold schemes and as such are useful only against decoders which can decrypt with probability greater than q, where q is a parameter.

The first scheme is the simplest one. It is an open scheme, based on “one-level” hash functions. Each hash function maps the n users into a set of $2k^2$ decryption keys. The keys themselves are kept secret, but the mapping (which user is mapped to what key) is publicly known. The personal key of every user consists of $O(k^2 \log n)$ decryption keys and this is also the number of decryptions that a user should perform in order to reveal the secret. The enabling block consists of $O(k^4 \log n)$ encrypted keys.

The second scheme is an open "two-level" scheme. This scheme is more complicated, but reduces the size of the enabling block by an $O(k/\log^3(k))$ factor. Here, a set of first-level hash functions maps the n users into a set of size k. Each function thereby induces a partition of the n users to k subsets. Each of these subsets is mapped separately by "second-level" hash functions into $\log^2 k$ decryption keys. This scheme requires $O(k^2 \log^2 k \log(n/k))$ keys and decryption operations per user, and an enabling block of $O(k^3 \log^4 k \log(n/k))$ encrypted keys.

The third scheme is a "one-level" secret scheme. Here, we assume that the hash functions, as well as the decryption keys, are kept secret. Being a secret scheme implies that the adversary does not know which keys correspond to any innocent user. There is a positive probability $p\,(0 < p < 1)$ that the adversary will be able to produce a pirate decoder which prevents the identification of any traitor. However, even if the keys known to the k collaborators enable the construction of such "wrongly incriminating" pirate decoders, choosing such a set is improbable. Furthermore, even if this unlikely event occurs, the adversary will not know that this is the case. The personal key in this scheme consists of $O(k \log(n/p))$ decryption keys (and the user should perform this number of decryptions). The enabling block has $O(k^2 \log(n/p))$ encrypted keys.

Our fourth scheme is a secret two-level scheme. Again, the saving in going from one level to two levels is in the size of the enabling blocks: The personal key and the number of required decryption operations are $O(\log(1/p) \log(n/p))$, and the enabling block contains $O(k \log(1/p) \log(n/p))$ encrypted keys. Compared to the previous scheme, the performance parameters are smaller by a factor of $k/\log(1/p)$, so this scheme is more efficient if $k \gg \log(1/p)$.

The last two schemes are threshold schemes, and as such are only good against pirate decoders which decrypt with probability greater than some predefined parameter q. The fifth scheme is a one-level threshold scheme. The personal key contains $m = \frac{4k}{3q} \log(n/p)$ decryption keys, which is of the same order as the key length in the one-level secret scheme (if q is constant, which is sufficient for most applications). The main improvement is in the data redundancy overhead which is only $4k$ encrypted keys and does not depend on n, and in requiring a user to perform only a single basic decryption operation in order to decrypt the secret.

The sixth scheme is a two-level threshold scheme, and it reduces the personal key length at the expense of slightly increasing the data redundancy overhead. The complexity depends on a parameter w. Define $b = \log(\frac{k}{p \log(1/p)})$. When w is constant (then the key length is minimal), the personal key is composed of $m = O(b \log(2n/p))$ decryption keys, the enabling block contains $O(k \log(k/qb))$ basic encryptions, and the user should perform $O(\log(k/b))$ decryptions.

a_1^0	a_1^1
a_2^0	a_2^1
$\vdots$	$\vdots$
$a_{\log n}^0$	$a_{\log n}^1$

Fig. 3. Keys for the 1-resilient scheme.

All schemes are constructed by choosing hash functions at random, and using probabilistic arguments to assert that the desired properties hold with overwhelming probability. These schemes are, therefore, not constructive (although the properties of the simplest scheme can be verified). We note, however, that there is no need to represent or store the whole function. It is only required that each user stores the outcome of the function, evaluated at the user's ID.

IV. Open Fully Resilient Schemes

A. A Simple Open One-Level Scheme

We describe in detail the first tracing scheme, starting with the simple case of a single traitor, $k = 1$. In this case, the data supplier generates $r = 2 \log n$ keys

$$\left\{a_1^0, a_1^1, a_2^0, a_2^1, \cdots a_{\log n}^0, a_{\log n}^1\right\}.$$

It is convenient to view these keys as organized in a matrix with $\log n$ rows and two columns (see Fig. 3).

Each user has a $\log n$ bit identity, and the personal key for user i is the set of $m = \log n$ keys

$$\left\{a_1^{b_1}, a_2^{b_2}, \cdots, a_{\log n}^{b_{\log n}}\right\}$$

where b_i is the ith bit in u's identity. Think of each personal key as selecting one key per row. Different users have at least one row where they differ in the selected keys.

The tracing scheme is used to encrypt a secret s. We always regard s as the key with which the cipher block can be decrypted. The data supplier encrypts s in the enabling block. A decoder typically first decrypts s from the enabling block and then uses s to decrypt the cipher block.

The secret s is encrypted in the enabling block as follows: It is split into $\log n$ secrets $s_1, s_2, \cdots, s_{\log n}$, i.e., the data supplier chooses at random $s_1, s_2, \cdots, s_{\log n}$ such that s is the bit-wise XOR of the s_i's. The value s_i is encrypted under the two keys of the ith row, a_i^0 and a_i^1. Both encryptions are added to the enabling block. Every user u can reconstruct all the s_i's and hence can decrypt s. On the other hand, any pirate decoder must contain a key for every row $i, 1 \le i \le \log n$ (otherwise, s_i would remain unknown and, consequently, s could not be obtained). Since at most one traitor is involved, the keys stored in the pirate decoder must be identical to the keys in the traitor's decoder. Therefore the pirate decoder uniquely identifies the single traitor.

$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,2k^2}$
$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,2k^2}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a_{\ell,1}$	$a_{\ell,2}$	$\cdots$	$a_{\ell,2k^2}$

Fig. 4. Keys for the simple k-resilient scheme.

When dealing with larger coalitions, we generalize the above scheme. We will now use matrices of keys with ℓ rows and $2k^2$ columns, where ℓ is a parameter to be specified later (see Fig. 4). The personal key of every user contains ℓ keys, one key per row. Again, a secret s is expressed as the bit-wise XOR of ℓ random s_i's. Each s_i is encrypted under all keys from the ith row. Therefore, to be able to find s with nonnegligible probability, a pirate decoder must contain one key from each row. So far the description is very similar to the previous 1-resilient scheme. The major difficulty we encounter is in the procedure for detecting traitors. Unlike the case $k = 1$, the pirate decoder might now contain keys from k different members of the coalition. It is, therefore, required to arrange the personal keys in such a way that the keys selected by each user are different from those selected by other users not only in a few rows, but in the vast majority of the rows. The best way we know of to achieve this goal is to assign keys to users in each row independently at random. In other words, each row is associated with a random hash function which chooses which entry (or column) is assigned to every user, and hash functions associated with different rows are independent. A detailed description of the scheme is given below.

Initialization: A set of ℓ hash functions $h_1, h_2, \cdots, h_\ell$ is chosen at random by the data supplier. Each hash function h_i maps $\{1, \cdots, n\}$ into the set $\{1, \cdots, 2k^2\}$. A set of $2k^2$ random keys is chosen for each row. The set

$$A_i = \{a_{i,1}, a_{i,2}, \cdots, a_{i,2k^2}\}$$

is assigned to the ith row. The personal key of user u is the set

$$P(u) = \{a_{1,h_1(u)}, a_{2,h_2(u)}, \cdots, a_{\ell,h_\ell(u)}\}.$$

Distributing a secret key: For each i $(i = 1, 2, \cdots, \ell)$ the data supplier encrypts a key s_i under each of the $2k^2$ keys in A_i. The final secret key s is the bit-wise exclusive–or of the "shares" s_i. Each authorized user has one key from every A_i, and can decrypt every s_i, and thus compute s.

Parameters: The memory required per user is $m = \ell$ keys. An enabling block to encode the secret key s consists of $2k^2$ encryptions of each s_i, totaling $r = 2k^2\ell$ encrypted keys.

Tracing: Assume that a pirate decoder $\mathcal{A}$ decrypts the content with probability q'. Let F be the set of locations in the matrix containing the keys which are known to the pirate who created $\mathcal{A}$. We can consider the keys in F as being part of the input to $\mathcal{A}$. If F contains at least one key from every row then it is possible to perform the "detection of traitors" process that is described in the sequel. Otherwise, we claim that that the encryption scheme which is used is insecure against a simple decoder $\mathcal{A}'$ which does not contain any key, uses $\mathcal{A}$ as a black box, and can also decrypt the content with probability q'. We were only able to prove the reduction from $\mathcal{A}'$ to $\mathcal{A}$ for a tracing scheme which is very similar to the scheme we presented: the only difference is that the scheme does not directly encrypt the cipher block with the secret s but rather with a value $s \oplus s'$, where s' is sent (in the clear) in the enabling block. The receiver decrypts the value s from the enabling block, calculates its exclusive–or with s', and uses the result to decrypt the cipher block. For simplicity, we present throughout the paper schemes which do not use the parameter s'. However, these schemes can be replaced by schemes which use s', for which we can prove a reduction. The overhead of these schemes is only negligibly greater than the overhead of the presented schemes.

The reduction: Given a decoder $\mathcal{A}$ which operates against a scheme with ℓ rows while containing keys from a set F of entries in at most $\ell - \ell'$ rows, we construct a decoder $\mathcal{A}'$ which operates against a scheme with ℓ' rows and does not contain any key. The input to $\mathcal{A}'$ is an enabling block (containing a value s') and a cipher block. The decoder chooses $\ell - \ell'$ random values $r_1, \cdots, r_{\ell-\ell'}$. For each value r_i it chooses $2k^2$ random keys and generates a row which contains $2k^2$ encryptions of r_i, one with each of the keys. It generates an enabling block which contains these rows, the rows of the enabling block which it received as input, and a value $s'' = s' \oplus (\oplus_{i=1}^{\ell-\ell'} r_i)$. The decoder $\mathcal{A}'$ inputs to $\mathcal{A}$ this enabling block, the cipher block, and the keys that $\mathcal{A}$ expects to receive (taken from a set F in the $\ell - \ell'$ rows that $\mathcal{A}'$ generated). The input to $\mathcal{A}$ is a valid encryption of the content that is encrypted in the cipher block (with the same distribution of keys as an original input to $\mathcal{A}$). Therefore, $\mathcal{A}$ (and hence $\mathcal{A}'$) will succeed in decrypting it with probability q'.

From here on we assume that the decoder contains at least one key from every row. Upon confiscation of a pirate decoder, at least one key from every set A_i (row) is exposed. We claim that it suffices to experiment with the decoder for this purpose, and it is not necessary to take the decoder apart ("reverse-engineer it"). The only assumption we should make is that it is possible to experiment with the decoder box and then reset it to its initial configuration. The proceudre that extracts the keys operates as follows. For all $1 \le i \le \ell$ and $0 \le j \le 2k^2$ perform the experiment E_{ij}: Prepare a normal encryption session, but instead of encrypting the key s_i with keys $\{a_{i,1}, \cdots a_{i,j}\}$ provide j random strings. Let $f_{i,j}$ be the fraction of times the box decrypts correctly on experiment E_{ij}. By assumption $f_{i,0}$ is nonnegligible (or the box is useless, since E_{i0} is its "normal" execution), and $f_{i,2k^2}$ is negligible since the key s_i is completely missing. There must be a $1 \le j \le 2k^2$ such that

$$f_{i,j} - f_{i,j-1} > (f_{i,0} - f_{i,2k^2})/2k^2.$$

For this j it can be deduced that $a_{ij} \in F$.

The set F contains at least ℓ keys (at least one key per set A_i). For each i, denote by $a_{i,t(i)}$ the key with minimal second subscript in $A_i \cap F$. The users in $h_i^{-1}(t(i))$ are those who could contribute this key to the pirate decoder. All the users in this set are identified and marked. This set includes at least one traitor, and possibly some innocent users. A count of all marks per user (for $i = 1, 2, \cdots, \ell$) is carried out. The user who has the largest number of marks (and this number must be at least ℓ/k) is declared to be the suspected traitor.

Goal: We show that there is a choice of hash functions, such that for all coalitions of size k and all pirate decoders they construct, the suspect is never an innocent user. Clearly, at least one of the traitors contributes at least ℓ/k of the keys $a_{i,t(i)}$. We will show that the probability (over all choices of hash functions) that an innocent user is marked ℓ/k times is negligible. This will prove the existence of hash functions with the desired properties.

Consider a specific user, say user 1, and a specific coalition T of k traitors (which does not include user 1). As hash functions are chosen at random, the value $h_i(1)$ is uniformly distributed in $\{1, \cdots, 2k^2\}$, and so the key $a_{i,h_i(1)}$ is uniformly distributed in A_i. The coalition gets at most k keys in A_i (out of the total $2k^2$). The probability that $a_{i,h_i(1)}$ is among these keys is, at most, $1/2k$.

Let X_i be a zero–one random variable, where $X_i = 1$ if $\exists u \in T$ s.t. $h_i(u) = h_i(1)$. The mean value of $\sum_{i=1}^{\ell} X_i$ is $\ell/2k$, and $\sum_{i=1}^{\ell} X_i$ is not smaller than the number of marks user 1 gets. If $\sum_{i=1}^{\ell} X_i < \ell/k$, then user 1 is not exposed as a suspect, since at least one traitor gets at least ℓ/k marks. We use the following version of Chernoff bound (see [2, Theorem A.12]) to bound the probability that $\sum_{i=1}^{\ell} X_i \geq \ell/k$. Let $X_1, \cdots, X_\ell$ be mutually independent random variables, with

$$\Pr[X_j = 1] = p$$
$$\Pr[X_j = 0] = 1 - p.$$

Then, for all $\beta \geq 1$

$$\Pr\left[\frac{1}{\ell}\sum_{j=1}^{\ell} X_j \geq \beta p\right] < \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^{p\ell}.$$

In our case, substituting $p = 1/2k$ and $\beta = 2$, we have

$$\Pr\left[\frac{1}{\ell}\sum_{i=1}^{\ell} X_i \geq \frac{1}{k}\right] < \left(\frac{e}{4}\right)^{\ell/2k} < 2^{-\ell/4k}.$$

The last bound considers one specific coalition and one specific innocent user. We demand that for a random scheme the expected number of pairs of a coalition and an innocent user, for which the coalition might frame the user, is less than 1. Then there exists a scheme in which no coalition can frame a user. We should take ℓ satisfying

$$n \cdot \binom{n}{k} \cdot 2^{-\ell/4k} < 1.$$

It suffices to take $\ell > 4k^2 \log n$. With this parameter, there is a choice of ℓ hash functions such that for every coalition and every authorized user not in the coalition, the innocent user is not incriminated by the tracing algorithm. We summarize the result in the next theorem.

Theorem 1: There is an open fully k-resilient traceability scheme, where a user's personal key consists of $m = 4k^2 \log n$ decryption keys, an enabling block consists of $r = 8k^4 \log n$ key encryptions, and a user should perform $4k^2 \log n$ decryptions in order to reveal the secret.

The discussion above shows the existence of open k resilient traceability schemes, and provides a randomized method for constructing a scheme that works with high probability. Although the theorem does not suggest an explicit construction, the desired properties of a given construction can be verified efficiently. The idea is to examine all the pairs of users (u, v) and check the number of functions h_i such that $h_i(v) = h_j(u)$. If this number is smaller than ℓ/k^2 then we can conclude that no coalition T of at most k users "covers" more than a $1/k$ fraction of the keys of u, and hence cannot incriminate u (this property is stronger than the property required for the scheme).

By considering pairwise differences, we can phrase the construction problem as a problem in coding theory (see [20]): construct a code with n codewords with length ℓ, over an alphabet of size $2k^2$, such that the distance between every two codewords is at least $\ell - \ell/k^2$. The goal is to construct such a code with as small ℓ as possible. There are no known explicit constructions that match the probabilistic bound. For the best known construction see [1] and references therein. For small k, the constructions of [1] yield a scheme with $m = O(k^6 \log n)$ and $r = O(k^8 \log n)$.

B. An Open Two-Level Scheme

The "two-level" traceability scheme, described in this subsection, can be thought of as iterating the previous construction two times. While it is more complicated than the simple scheme, it saves a factor of about k in the broadcast overhead.

Theorem 2: There is an open fully k-resilient traceability scheme, where a user's personal key consists of $m = \frac{8}{3}k^2 \log^2 k \log(en/k)$ decryption keys, and an enabling block consists of $r = \frac{32}{3}ek^3 \log^4 k \log(en/k)$ key encryptions. A user should perform $\frac{8}{3}k^2 \log^2 k \log(en/k)$ decryptions in order to decrypt the secret.

Proof: As in the simple scheme, the proof is existential (but here we do not know how to efficiently verify that a given scheme is "good"). It will be convenient to view the keys as organized in ℓ blocks. Each block is a d-by-$\lceil ek \rceil$ matrix, where ℓ and d are parameters that will be specified later. It is important to note that each entry in the matrix contains $4\log^2 k$ keys (see Fig. 5). For each block, every user gets one key per row. All these d keys are taken from the same column.

The secret key s is constructed in a way that forces any decoder to satisfy the following constraint: For each block, there exists a column, such that for every row, the decoder contains a key from the entry at the intersection of the column and the row. In other words, the decoder contains d keys from a certain column in every block. We now describe the system in detail.

Initialization: A set of ℓ "first-level" hash functions $h_1, h_2, \cdots, h_\ell$, each mapping $\{1, \cdots, n\}$ to $\{1, \cdots, \lceil ek \rceil\}$, is chosen independently at random. The function h_i is used to map the users into the columns of the ith block. For each block i $(i = 1, 2, \cdots, \ell)$ and each row j $(1 \leq j \leq d)$, a "second-level" hash function $g_{i,j}$, which maps $\{1, \cdots, n\}$ to $\{1, \cdots, 4\log^2 k\}$, is chosen independently at random. The function $g_{i,j}$ is used to map users into specific elements in the entries of the jth row of

$a_{1,1,1,1},\cdots,a_{1,1,1,4\log^2 k}$	$a_{1,1,2,1},\cdots,a_{1,1,2,4\log^2 k}$	$\cdots$	$a_{1,1,\lceil ek\rceil,1},\cdots,a_{1,1,\lceil ek\rceil,4\log^2 k}$
$a_{1,2,1,1},\cdots,a_{1,2,1,4\log^2 k}$	$a_{1,2,2,1},\cdots,a_{1,2,2,4\log^2 k}$	$\cdots$	$a_{1,2,\lceil ek\rceil,1},\cdots,a_{1,2,\lceil ek\rceil,4\log^2 k}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a_{1,d,1,1},\cdots,a_{1,d,1,4\log^2 k}$	$a_{1,d,2,1},\cdots,a_{1,d,2,4\log^2 k}$	$\cdots$	$a_{1,d,\lceil ek\rceil,1},\cdots,a_{1,d,\lceil ek\rceil,4\log^2 k}$

$a_{2,1,1,1},\cdots,a_{2,1,1,4\log^2 k}$	$a_{2,1,2,1},\cdots,a_{2,1,2,4\log^2 k}$	$\cdots$	$a_{2,1,\lceil ek\rceil,1},\cdots,a_{2,1,\lceil ek\rceil,4\log^2 k}$
$a_{2,2,1,1},\cdots,a_{2,2,1,4\log^2 k}$	$a_{2,2,2,1},\cdots,a_{2,2,2,4\log^2 k}$	$\cdots$	$a_{2,2,\lceil ek\rceil,1},\cdots,a_{2,2,\lceil ek\rceil,4\log^2 k}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a_{2,d,1,1},\cdots,a_{2,d,1,4\log^2 k}$	$a_{2,d,2,1},\cdots,a_{2,d,2,4\log^2 k}$	$\cdots$	$a_{2,d,\lceil ek\rceil,1},\cdots,a_{2,d,\lceil ek\rceil,4\log^2 k}$

$$\vdots$$

$a_{\ell,1,1,1},\cdots,a_{\ell,1,1,4\log^2 k}$	$a_{\ell,1,2,1},\cdots,a_{\ell,1,2,4\log^2 k}$	$\cdots$	$a_{\ell,1,\lceil ek\rceil,1},\cdots,a_{\ell,1,\lceil ek\rceil,4\log^2 k}$
$a_{\ell,2,1,1},\cdots,a_{\ell,2,1,4\log^2 k}$	$a_{\ell,2,2,1},\cdots,a_{\ell,2,2,4\log^2 k}$	$\cdots$	$a_{\ell,2,\lceil ek\rceil,1},\cdots,a_{\ell,2,\lceil ek\rceil,4\log^2 k}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a_{\ell,d,1,1},\cdots,a_{\ell,d,1,4\log^2 k}$	$a_{\ell,d,2,1},\cdots,a_{\ell,d,2,4\log^2 k}$	$\cdots$	$a_{\ell,d,\lceil ek\rceil,1},\cdots,a_{\ell,d,\lceil ek\rceil,4\log^2 k}$

Fig. 5. Keys for the two-level scheme.

the ith block (a user is assigned to a certain column of a block and is always mapped to elements in this column).

Every user $u \in \{1,\cdots,n\}$ receives ℓd keys, d keys per block. The keys are

$$a_{1,1,h_1(u),g_{1,1}(u)}, a_{1,2,h_1(u),g_{1,2}(u)}, \cdots, a_{1,d,h_1(u),g_{1,d}(u)}$$

(these d keys are all from column $h_1(u)$ of the first block) through

$$a_{\ell,1,h_\ell(u),g_{\ell,1}(u)}, a_{\ell,2,h_\ell(u),g_{\ell,2}(u)}, \cdots, a_{\ell,d,h_\ell(u),g_{\ell,d}(u)}$$

(these d keys are all from column $h_\ell(u)$ of the ℓ-th block).

Distributing a secret key: The data supplier chooses at random ℓ independent keys (shares) $s_1,\cdots,s_\ell$. The secret key is $s = \mathrm{XOR}_{i=1}^{\ell} s_i$.

Each s_i is divided into $d \cdot \lceil ek\rceil$ shares that correspond to the entries in the ith block, one share per an intersection of a row and a column. These shares are random subject to the constraint that the exclusive–or of the shares of each column is equal to s_i. That is, if we denote the share of the entry in row j $(j = 1,\cdots,d)$ and column c $(c = 1,\cdots,\lceil ek\rceil)$ in block i $(i = 1,2,\cdots\ell)$ as $s_{i,j,c}$, the shares satisfy

$$\begin{aligned} s_i &= \mathrm{XOR}(s_{i,1,1},\cdots,s_{i,d,1}) \\ &= \mathrm{XOR}(s_{i,1,2},\cdots,s_{i,d,2}) \\ &\vdots \\ &= \mathrm{XOR}(s_{i,1,\lceil ek\rceil},\cdots,s_{i,d,\lceil ek\rceil}). \end{aligned}$$

The encryptions of the share $s_{i,j,c}$ under each of the $4\log^2 k$ keys in the entry (j,c) of the ith block are added to the enabling block. To find the key s one needs all the shares s_i. Therefore, for every block i there should be one column c such that for each row j at least one key from the entry (j,c) is in the decoder.

User u has the d keys

$$a_{i,1,h_i(u),g_{i,1}(u)}, a_{i,2,h_i(u),g_{i,2}(u)}, \cdots, a_{i,d,h_i(u),g_{i,d}(u)}$$

in his personal key. They enable him to decrypt and find

$$s_{i,1,h_i(u)},\cdots,s_{i,d,h_i(u)}$$

which make it possible to reconstruct each s_i and then compute the secret key s.

*Parameters:*The personal key consists of $m = \ell d$ keys. The total number of key encryptions in an enabling block is $4ek\ell d \log^2 k$.

Tracing: Assume that a pirate decoder $\mathcal{A}$ decrypts the content with probability q'. If it holds for every block $1 \leq i \leq \ell$ that there exists a column c such that $\mathcal{A}$ contains a key from the intersection of c with each of the rows, then the *detection of traitors* process that is described in the next paragraph detects at least one of the traitors. Otherwise, the bit sensitivity of the XOR operation guarantees that it is possible to construct a decoder $\mathcal{A}'$ which uses $\mathcal{A}$ and is able to decrypt the content with probability q', even without knowing any of the keys.

Upon confiscation of a pirate decoder, we assume here that it stores a subset F of the keys which for every block contains keys from the intersection of one column with all the rows. The subset is exposed using the following procedure for every block i and column c: Let $M_{0,0}^{i,c}$ be an enabling block in which the encryptions with the keys of all the columns of block i, except column c, are replaced with random data. For every row $1 \leq j \leq d$ and every entry $1 \leq s \leq 4\log^2 k$ build $M_{j,s}^{i,c}$ from $M_{0,0}^{i,c}$ by replacing with random data the encryptions in block i which are in the first s positions in the entry (j, c). There is at least one c such that $M_{0,0}^{i,c}$ enables decryption with nonnegligible probability, whereas for every row j the enabling block $M_{j,s}^{i,c}$ allows correct decryption with negligible probability. Let $t_j^{i,c}$ be the location of the key that caused the maximum decrease in the decryption probability for the jth row in a column c of block i. Then *mark* the users u s.t. $h_i(u) = c$ and $g_{i,j}(u) = t_j^{i,c}$. All users who are marked at least $d/\log k$ times for block i are suspects for s_i. The user who is a suspect for the largest number of s_i's is identified as a potential traitor.[2]

Goal: We want to show that there is a choice of hash functions such that for all coalitions, an innocent user is never identified as a traitor.

Consider a specific user, say user 1, and a specific coalition T of k traitors (which does not include user 1). We first bound the probability that user 1 will be a suspect for s_i. The first level hash function h_i partitions the users to $\lceil ek \rceil$ subsets $\{h_i^{-1}(1), \cdots, h^{-1}(\lceil ek \rceil)\}$. The expected maximum number of traitors in these $\lceil ek \rceil$ subsets is $\log k / \log\log k$. The probability that user 1 is hashed to a subset together with more than $\log k$ traitors is, at most,

$$\binom{k}{\log k} \cdot (ek)^{-\log k} \leq \left(\frac{ek}{\log k}\right)^{\log k} (ek)^{-\log k} = \left(\frac{1}{\log k}\right)^{\log k} \underbrace{<}_{\text{if } k \geq 16} \frac{1}{16k}.$$

Denote $h_i(1) = c$. Consider the conditional probability space where $T \cap h_i^{-1}(c)$ indeed contains at most $\log k$ traitors. In this conditional space, the d keys

$$a_{i,1,h_i(u_1),g_{i,1}(u_1)}, a_{i,2,h_i(u_2),g_{i,2}(u_2)}, \cdots, a_{i,d,h_i(u_d),g_{i,d}(u_d)}$$

in the pirate decoder come from the personal keys of $T \cap h_i^{-1}(c)$. The tracing algorithm can mark user 1 with respect to the jth row in the block i if there is some $u \in T \cap h_i^{-1}(c)$ such that $g_{i,j}(1) = g_{i,j}(u)$. The range of $g_{i,j}$ contains $4\log^2 k$ elements. At most $\log k$ of these are in $g_{i,j}(T \cap h_i^{-1}(c))$. So the probability that user 1 is marked with respect to the jth row in block i is, at most, $1/(4\log k)$. The expected number of times user 1 will be marked, with respect to the d functions $g_{i,1}, \cdots, g_{i,d}$, is, at most, $d/(4\log k)$. We use the Chernoff bound to estimate the probability that user 1 is a suspect for s_i.

Set $X_j = 1$ if user 1 is marked with respect to the jth row in block i, and $X_j = 0$ otherwise. Then

$$\Pr[X_j = 1] \leq 1/(4\log k).$$

By the Chernoff bound, with $p = 1/(4\log k)$ and $\beta = 4$

$$\Pr\left[\frac{1}{d}\sum_{j=1}^{d} X_j \geq \frac{1}{\log k}\right] < \left(\frac{e^3}{4^4}\right)^{d/(4\log k)} \leq 2^{-3d/(4\log k)}.$$

Setting $d = (8/3)\log^2 k$, the conditional probability that user 1 is a suspect for s_i is, at most, $2^{-2\log k} < 1/16k$ (when $k \geq 16$). The probability of the condition (at most $\log k$ traitors mapped together with 1 by the function h_i) not happening is, at most, $1/16k$. So overall, the total (unconditional) probability that user 1 is the suspect for s_i is, at most, $1/(8k)$.

Let us check the probability that user 1 is the suspect for at least $(3\ell/4k)$ of the blocks. For $i = 1, \cdots, \ell$, let $Y_i = 1$ if user 1 is the suspect for s_i, and $Y_i = 0$ otherwise. Then

$$\Pr\left[\frac{1}{\ell}\sum_{i=1}^{\ell} Y_i \geq \frac{3}{4k}\right] < \left(\frac{e^5}{6^6}\right)^{\ell/8k} < 2^{-\ell/k}.$$

So with probability at least $1 - 2^{-\ell/k}$, user 1 is a suspect for fewer than $3\ell/4k$ of the s_i.

Denote a block as *bad* if it contains a column into which $\log k$ or more traitors have been mapped, and *good* otherwise. In a good block at least one of the traitors is declared a suspect. Denote by ℓ' the number of good blocks. Next we show that the probability that $\ell' < \frac{3\ell}{4}$ is small. We previously showed that the probability that $\log k$ or more traitors are mapped to the same column is at most $1/16k$, namely, the probability that a block is bad is at most $1/16$. For $i = 1, \cdots, \ell$ let $Z_i = 1$ if block i is bad. Then the probability that there are $\ell/4$ bad blocks is at most

$$\Pr\left[\frac{1}{\ell}\sum_{i=1}^{\ell} Z_i \geq \frac{1}{4}\right] < \left(\frac{e^3}{4^4}\right)^{\ell/16} < 2^{-\ell/5}.$$

For every good block i, at least one member of T is a suspect for s_i because in each row at least one of them is marked. T contains k traitors, and so there must be one or more traitors who is a suspect for at least $\ell'/k > \frac{3\ell}{4k} s_i$'s. Therefore, the probability that user 1 is mistakenly identified as a traitor in this case is smaller than $2^{-\ell/k}$. Note that the definition of a *good* or a *bad* block does not depend on the user's identity (but of course does depend on T). Therefore, the probability that for one of the $\binom{n}{k}$ possible coalitions of size k, and given that there are at least $3\ell/4$ good blocks, some good user is mistakenly identified, is smaller than $n \cdot \binom{n}{k} \cdot 2^{-\ell/k}$. The probability that for some coalition there are less than $3\ell/4$ good blocks is at most $\binom{n}{k} 2^{-\ell/5}$. Setting

[2]Note that this procedure relies on the fact that for each block there is a column c such that the pirate decoder continues to decrypt even if we corrupt all the entries in the enabling block that correspond to all the keys from the block which do not come from column c. However, even if it is not the case, we can search for a minimum set of columns for which this property holds, and then the following analysis still holds.

$\ell = k^2 \log(en/k)$, the total probability is smaller than 1. This means that there exists a choice of hash functions h_i and $g_{i,j}$ such that a good user is never mistakenly identified as a traitor. The resulting open k-traceability scheme (which is good for any $k \geq 16$) has parameters

$$m = \ell d = \frac{8}{3} k^2 \log^2 k \log(en/k)$$

and

$$r = 4ek\ell d \log^2 k = \frac{32}{3} ek^3 \log^4 k \log(en/k). \qquad \square$$

V. Secret Fully Resilient Schemes

Secret schemes can be made more efficient than open schemes since the 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 install in a pirate decoder keys that incriminate a specific user.

A. A Secret One-Level Scheme

The first proposed scheme is one-level. The major source of saving is that it suffices to map the n users into a set of $4k$ keys (rather than the set of size $2k^2$ of the open one-level scheme). A coalition of size k will contain the key of any specific user with constant probability. However, as the traitors do not know which key this is, any key they choose to insert into the pirate decoder will miss (with high probability) the key of this user.

Initialization: There are n users, each with a unique identity $u \in \{1, 2, \cdots, n\}$. Let ℓ be a parameter. A set of ℓ hash functions $h_1, h_2, \cdots, h_\ell$ are chosen independently at random. Each hash function h_i maps $\{1, \cdots, n\}$ into a set of $4k$ random keys $A_i = \{a_{i,1}, a_{i,2}, \cdots, a_{i,4k}\}$. The hash functions are kept secret as well. User u receives, upon initialization, the indices and values of ℓ keys $\{a_1, h_1(u), \cdots, a_\ell, h_\ell(u)\}$.

Distributing a key: For each i $(i = 1, 2, \cdots, \ell)$ the data supplier encrypts a random s_i under each of the $4k$ keys in A_i. The final key is the bit-wise exclusive–or of the s_i's. Each authorized user has one key from A_i, so he can decrypt every s_i and compute s.

Parameters: The memory required per user is $m = \ell$ keys. The data redundancy overhead used in distributing the key s is $r = 4k\ell$.

Tracing: As was shown for the one-level open scheme, a pirate decoder must contain a key from every row if the encryption schemes that are used are secure. We show next how to trace the traitors given a decoder which contains a key from every row.

Upon confiscation of a pirate decoder, a set of keys contained in it, F, is extracted using the methods we have described for the one-level open scheme. F contains ℓ keys, one per set A_i. Denote by $f_i \in A_i$ the key in $F \cap A_i$. The tracing algorithm knows the values of the functions h_i and, therefore, can identify and mark for each i the users in $h_i^{-1}(f_i)$. The user with the largest number of marks is exposed.

Goal: We want to show that for all coalitions, the probability of exposing a user who is not a traitor is negligible. Clearly, at least one of the traitors contributes at least ℓ/k of the keys to the pirate decoder. It should be shown that the probability that a good user is marked ℓ/k times is negligible. Consider a specific user, say user 1, and a specific coalition T of k traitors (which does not include user 1). As the hash functions are random the value $a_i = h_i(1)$ is uniformly distributed in A_i, even given the k values hashed by h_i from the names of the coalition members. The probability that the value f_i extracted from the pirate decoder equals a_i is therefore $1/4k$. Let X_i be a zero–one random variable, where $X_i = 1$ if $a_i = f_i$. The mean value of $\sum_{i=1}^{\ell} X_i$ is $\ell/4k$. By the version of Chernoff bound used in Section IV-A (see [2, Theorem A.12])

$$\Pr\left[\frac{1}{\ell}\sum_{i=1}^{\ell} X_i \geq 4 \cdot \frac{1}{4k}\right] < \left(\frac{e^3}{4^4}\right)^{\ell/4k} < 2^{-3\ell/4k}.$$

We choose ℓ satisfying $n \cdot 2^{-3\ell/4k} < p$. That is, $\ell > 4k\log(n/p)/3$. Then for every coalition it holds that the probability that it can frame an innocent user is at most p. The following theorem sums the construction.

Theorem 3: There is a fully (p,k)-resilient secret traceability scheme, where a user's personal key consists of $m = 4/3 \cdot k\log(n/p)$ decryption keys, and an enabling block consists of $16/3 \cdot k^2 \log(n/p)$ key encryptions. A user should perform $4/3 \cdot k\log(n/p)$ decryptions in order to decrypt the secret.

B. A Secret Two-Level Scheme

A two-level scheme improves the performance of the one-level scheme of the previous section whenever $k \gg \log 1/p$. The difference between this scheme and the open two-level scheme in Section IV-B is that here it is sufficient to use only one mapping at the first level and hope that it is successful (which happens with good probability), whereas the two-level open scheme used two mappings. In the Appendix we present a somewhat simpler two-level secret scheme, which achieves slightly less efficient performance.

The basic idea of the construction is to randomly map the users into a small range, such that the probability of mapping together more than a small threshold b of traitors is smaller than $p/2$. An independent tracing scheme (secure with probability $p/2$ against b traitors) is employed for every value in the range. The overall error probability is, therefore, at most p.

The construction uses a random mapping h from the domain $\{1, \cdots, n\}$ to a range of size $2ek/b'$, where $b' = b - \frac{b}{b-1}\ln(ek/b)$. Then for any fixed set of k traitors, the probability that b or more traitors are mapped together by h is, at most,

$$\begin{aligned}\binom{k}{b}\left(\frac{b'}{2ek}\right)^{b-1} &\leq \left(\frac{ek}{b}\right)^{b}\left(\frac{b - \frac{b}{b-1}\ln(ek/b)}{2ek}\right)^{b-1}\\ &= \frac{ek}{b2^{b-1}}\left(1 - \frac{\ln(ek/b)}{b-1}\right)^{b-1}\\ &\approx \frac{ek}{b2^{b-1}} e^{-\ln(ek/b)} = \frac{1}{2^{b-1}}.\end{aligned}$$

Setting $b = \log(4/p)$ implies that this probability is, at most, $p/2$. Once such a mapping is chosen we continue by constructing the secret $(b, p/2)$-resilient one-level scheme of Section V-A 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.

The detection of traitors is performed as follows: Assume that a pirate decoder $\mathcal{A}$ contains keys from all the rows of a certain subscheme (otherwise it is possible to build a decoder $\mathcal{A}'$ which contains no key and can decrypt the content with the same probability as $\mathcal{A}$). First, it is required to identify a subschceme for which the decoder contains a key from every row. Then these keys have to be identified, and the source of these keys can be found by the same methods that were used for the one-level scheme. To perform this, prepare a valid encrypted message M_0 and choose a random order of the subschemes. In step i, construct the message M_i by replacing with random data the parts of the message M_{i-1} which are encrypted by the keys of the ith subscheme (in the chosen order). Feed the message M_i into the pirate decoder. The message M_0 is a valid message and a pirate decoder should decrypt it with high probability, whereas the message $M_{2ek/b'}$ contains only random data and therefore cannot be decrypted with nonnegligible probability. Let M_i be the message that caused the maximum decrease in the decryption probability. The decryption keys of subscheme i must be stored in the decoder. Now, start from the message M_{i-1} and change the keys of the ith subscheme according to the key extraction procedure that was described for the one-level scheme, and find a set containing one key from every row of the ith subscheme, which is contained in the pirate decoder.

Assume that there is no subscheme into which more than b traitors are mapped together (an event which happens with probability at least $1 - p/2$). Then the conditional probability of incriminating any of the innocent users in subscheme i (in which we search for traitors in the process we described), is the probability that the subset of traitors that is mapped to i (and by assumption is of size at most b) manages to incriminate an innocent user. Since each subscheme is $(b, p/2)$-resilient, this probability is, at most, $p/2$. The unconditional probability that there is a user who is wrongly incriminated is, therefore, at most p.

The number of keys a user gets in this scheme is simply the number of keys a user gets in the $(b, p/2)$-resilient scheme, that is, $m = \frac{4}{3}\log(4/p)\log(2n/p)$. The size of the enabling block is $2ek/b'$ times the size of the enabling block in the $(b, p/2)$-resilient scheme, i.e.,

$$\begin{aligned}&\frac{32}{3}ek\log(2n/p)\frac{b^2}{b'}\\&\quad=\frac{32}{3}ekb\log(2n/p)\frac{1}{1-\frac{1}{b-1}\ln(ek/b)}\\&\quad=\frac{32}{3}ekb\log(2n/p)\left(1+\frac{\ln(ek/b)}{b-1-\ln(ek/b)}\right).\end{aligned}$$

We thus obtain the following theorem:

Theorem 4: There is a fully (p, k)-resilient secret traceability scheme, where a user's personal key consists of $m = \frac{4}{3}b\log(2n/p)$ decryption keys, and an enabling block consists of at most

$$\frac{32}{3}ekb\log(2n/p)\left(1+\frac{\ln(ek/b)}{b-1-\ln(ek/b)}\right)$$

encryptions, where $b = \log(4/p)$. A user should perform $\frac{4}{3}b\log(2n/p)$ decryptions in order to decrypt the secret.

Note that unless k is very large compared to $1/p$, the last multiplicand

$$\left(1+\frac{\ln(ek/b)}{b-1-\ln(ek/b)}\right)$$

is small. For example, it is smaller than 4 if $p = 1/100$ and $k < 1000$, or if $p = 1/1000$ and $k < 16500$.

We can get slightly better results if we consider the fact the each subscheme should handle fewer users. The expected number of users that are mapped to a certain subscheme is about $n' = \frac{nb'}{2ek}$, and the probability that the number of users that are mapped to a certain subscheme is much larger than n' is small. Therefore, the subschemes can be designed for $O(n')$ users only, resulting in lower complexity.

VI. Threshold Schemes

The performance guarantee of fully resilient tracing schemes might be an overkill for many applications. Fully resilient schemes trace the source of keys of any decoder which uses a secure encryption function and decrypts with nonnegligible probability. In many applications, it is obvious that pirates cannot sell pirate decoders which do not decrypt with probability which is very close to 1 (e.g., decoders for TV transmissions). For such applications, it is possible to design tracing schemes which only trace the source of keys of decoders which decrypt with high probability (and do not necessarily perform well against decoders which decrypt with lower probability). This section introduces such schemes which are more efficient than fully resilient schemes (refer to Table I for a comparison between the complexity of different threshold schemes and the most efficient fully resilient scheme).

Recall Definition 1. It is assumed that the basic encryption scheme cannot be decrypted with probability better than q'' without using the decryption keys. Fully resilient schemes are designed to trace the source of keys of any decoder which decrypts with probability better than $q' = q''$. The target of q-threshold schemes is to trace the source of keys of any decoder which decrypts with probability better than $q' = q + q''$. The parameter q is the advantage of a pirate decoder $\mathcal{A}$ in decrypting messages, over the success probability of a decoder $\mathcal{A}'$ which does not contain any of the decryption keys. Fully resilient schemes are designed to trace for any $q > 0$. However, since the probability q'' is assumed to be negligible, it can be assumed that q must be large in order for a pirate decoder to be useful.

The complexity of q-threshold schemes depends on the value of the parameter q: They are more efficient for larger values of q. The schemes are secret in the sense that the set of keys that each user receives is unknown to other users.

The benefit of using threshold tracing schemes is a reduction in the data redundancy overhead and in the number of decryptions that the receiver should perform, whereas the length of the personal key is almost as short as in secret fully resilient schemes. A one-level threshold scheme results in a very short data redundancy overhead, and requires the receiver to perform a single decryption operation. The key is only marginally longer than in the secret one-level scheme of Section V-A. This is also the case with two-level threshold schemes, although compared to the one-level threshold schemes the key is longer. In particular, the two-level threshold scheme of Section VI–B2 achieves better efficiency than the best fully resilient scheme (of Section V-B) in *all* complexity parameters.

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 these two complexity measures (for instance, the last entry of Table I demonstrates a reasonable such tradeoff).

A. A One-Level Threshold Scheme

The basic scheme is similar to the one-level secret scheme with the following exception: the secret s is not divided into ℓ shares but rather into t shares (where $(t < \ell)$ is a parameter) which are encrypted using t rows chosen uniformly at random. These rows are chosen independently for every enabling block, and their indices are sent at the beginning of the block so that the decoder can know which keys to use. A legitimate user has a key from every row and can therefore recover s. However, if a pirate decoder does not contain a key from each of the t rows it cannot obtain s. The data redundancy overhead is composed of encryptions with the keys of the t rows and, in addition, the names of the t rows which were chosen. Note that the decryption process now requires less operations from the receivers, they should perform only t decryptions, instead of ℓ decryptions in the fully resilient schemes.

In the one-level secret fully resilient scheme of Section V-A, each row contained $4k$ keys and setting the number of rows to be $\ell = \frac{4k}{3}\log(n/p)$ suffices to get a probability of at least $1-p$ for tracing the traitors. The threshold scheme depends on a parameter w (in the range $0 < w < 1$) such that it is possible to trace the source of keys if the pirate decoder contains keys from a fraction of at least w of the rows. The number of shares into which the secret is divided (the parameter t) is set such that if a decoder contains keys from a fraction of less than w of the rows, it cannot gain an advantage better than q in finding s. Therefore, a pirate decoder which gains an advantage which is better than q should contain a set with one key from at least $w\ell$ of the rows. In this case, at least one traitor contributes at least $\frac{w\ell}{k}$ of the keys in this set, and in comparison, an innocent user is expected to have only $\frac{w\ell}{4k}$ keys which are included in this set. The probability of tracing a traitor can be calculated using the same analysis as in the secret, fully resilient one-level scheme, substituting $w\ell$ instead of ℓ for the number of rows for which there is information. To obtain a (k,p)-resilient scheme it is enough to set the number of rows to $1/w$ the number of rows in the fully resilient scheme, that is $\ell = \frac{4k}{3w}\log(n/p)$.

Fix w, the fraction of rows that enables to trace a pirate. The parameter t is set to ensure that the probability that t random rows are all contained in a subset of $w\ell$ rows, is, at most, q. Therefore, in order to achieve decryption probability which is greater than q, the decoder must have keys from at least a fraction w of the rows. To set the value of t, based on the parameters w and q, observe that the probability that a pirate decoder which has keys from $w\ell$ rows, contains keys from t random rows is, at most, w^t, and therefore setting

$$t = \log_w q = \frac{\log(1/q)}{\log(1/w)}$$

suffices to make this probability at most q. For example, it is possible to set $w = q$, fix the number of rows accordingly, and then set $t = 1$. 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 $4k$.

Detection of traitors: It can be assumed that $\mathcal{A}$ contains keys from at least $w\ell$ rows, since otherwise it can be used to generate a decoder which does not contain any key and decrypts with probability at least q. It is possible to expose the keys which are contained in a confiscated decoder $\mathcal{A}$ by treating it as a black box, like with fully resilient schemes: Choose a random order of the entries of the matrix. Start with a valid message M_0. In step i take the message M_{i-1} and create the message M_i by replacing the data encrypted with the key of the ith entry (according to the chosen order) by random data. Feed the message M_i into the decoder. Let the set G_i contain the keys in the entries numbered $i+1$ and higher. Let M_j be the first message for which the pirate decoder contains keys from G_j in less than a w fraction of the rows. Then step j can be identified since in this step the probability with which the decoder can correctly decrypt reaches below q (and decreases by a factor of at least $(1-1/w)^t$). When this happens, conclude that the key corresponding to entry j is contained in the pirate decoder.[3] Repeat this procedure until you find a key from $w\ell$ rows. Choose one key from each row. Announce the user who contributed the maximum number of keys to this set (this number should be at least $w\ell/k$) to be a traitor.

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 w which is a parameter. As w increases the key length decreases and the data redundancy overhead increases. Then we state the results for $w = q$.

Theorem 5: 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

$$4k\log_w q = 4k\frac{\log(1/q)}{\log(1/w)}$$

keys. A user should perform $\frac{\log(1/q)}{\log(1/w)}$ in order to decrypt the secret.

[3]Note that in the detection process, it is not sufficient to change the entries of just a single row r_i and check in which one of these entries the probability decreases, since the decoder might contain more rows than are needed for the probability of decryption to be q, but still output the correct decryption result only with probability q. Then even when the data encrypted with the keys of row r_i is random, the decoder can still have a correct output with probability q by using the keys it has from the other rows, and it is impossible to decide which of the keys of row r_i in contained in the decoder.

If we set $w = q$ then a personal key consists of $\frac{4k}{3q}\log(n/p)$ keys, the data redundancy overhead is of only $4k$ keys, and a receiver should perform only a single decryption in order to reveal the secret.

The scheme we presented displays a tremendous improvement in the data redundancy overhead, but the personal key is quite long, its length is a little larger than in the fully resilient one-level secret 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.

B. Two-Level Threshold Schemes

Two-level threshold schemes are constructed from one-level threshold schemes in the same way fully resilient two-level secret schemes were constructed. We first present a basic construction which displays a tradeoff between the personal key length and the data redundancy overhead, and which can have 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, at the price of increasing a little the data redundancy.

1) The Basic Construction: The construction uses a random mapping $h:\{1,\cdots,n\}\rightarrow\{1,\cdots,(2ek/b)\}$. It constructs $2ek/b$ one-level subschemes secure against coalitions of b traitors and uses h to map each user to a subscheme. As with the fully resilient schemes, it is required that the probability that b or more of the k traitors are mapped together is less than $p/2$, namely, that

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

The inequality is satisfied when $b = \log\left(\frac{4ek}{p\log(1/p)}\right)$. 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 less than $\tilde{q} = \frac{qb}{2ek}$ from the success probability of an adversary who does not have any of the keys, or they 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 stages of the initialization and the distribution of the secrets 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 number of rows that enable decryption 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 procedure that extracts keys from a pirate decoder is performed in two stages. First, to find a suspicious subscheme, it starts with a valid message M_0, and as with the fully resilient secret two-level scheme, repeatedly changes all the information encrypted with the keys of the ith subscheme into random data. Let subscheme i be the subscheme for which the decryption success probability dropped the most. Subscheme i will be checked in the next stage: Start with the message M_{i-1} and apply to subscheme i the method used for tracing the sources of the keys of the one-level threshold scheme. If no more than b traitors are mapped together, then the suspect that is finally announced is a traitor with probability at least $1-p/2$. We therefore obtain the following theorem.

Theorem 6: There is a q-threshold (p,k)-resilient scheme, with the parameter w taking values in $\left[\frac{qb}{2ek},1\right)$, where

$$b = \log\left(\frac{4ek}{p\log(1/p)}\right)$$

in which

- the length of the personal key is $m = \frac{4}{3w}b\log(2n/p)$ basic keys;
- the data redundancy overhead is
$$8ek\log\left(\frac{2ek}{qb}\right)/\log(1/w)$$
basic encryptions;
- a receiver should perform $\log(2ek/(qb))/\log(1/w)$ 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, then, 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. When w is set to this value, the data redundancy overhead is minimized to $8ek$ 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 extreme, 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 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 be defined for a smaller set of users and then the length of the personal key is smaller.

2) Shorter Personal Keys: This section presents a threshold scheme which improves all the complexity parameters of the most efficient fully resilient scheme (whereas the previous tracing scheme had a great improvement in the data redundancy and decryption overheads, but increased the length of 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 now composed of $(4/3w)b_2\log(2n/p)$ keys, and the data redundancy overhead is of $8ek\frac{b_2}{b_1}\log\left(\frac{2ek}{qb_1}\right)/\log(1/w)$ basic encryptions.

The values b_1, b_2 should satisfy the following inequality:

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

Assume $b_2 = b_1^\alpha = b^\alpha$ $(\alpha > 1)$. The previous inequality is satisfied if

$$b \ge \sqrt[\alpha]{\frac{\alpha}{\alpha-1}\cdot\frac{\log(k/p)}{\log\log(k/p)}}.$$

We, therefore, obtain the following theorem.

Theorem 7: For every $\alpha > 1$ there is a q-threshold (p,k)-resilient scheme, with the parameter w taking values in $\left[\frac{qb}{2ek}, 1\right)$, 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

  $$8ekb^{\alpha-1}\log\left(\frac{2ek}{qb}\right)/\log(1/w)$$

  basic encryptions;
- a receiver should perform $\log(2ek/(qb))/\log(1/w)$ 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 $\alpha \to \infty$ are as follows.

- 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 data redundancy overhead is

  $$8ek\frac{\log(k/p)}{\log\log(k/p)}\log(2ek/q)/\log(1/w)$$

  basic encryptions.
- The number of decryptions that a receiver should perform is $\log(2ek/q)/\log(1/w)$.

This scheme has the shortest personal key among all the schemes we presented, but the data redundancy overhead is longer than in the basic two-level threshold scheme. However, the data redundancy is still shorter than in the fully resilient schemes.

VII. Lower Bounds for Open Schemes

In this section we derive lower bounds on the total number of keys r and on the number of keys per user m in any scheme that has the properties of our open schemes. Namely, schemes where the set of all keys is $S = \{s_1, s_2, \cdots s_r\}$, and each user i gets a subset $U_i \subset S$ of size m. We require that no coalition of k users ("traitors") $\{i_1, i_2, \cdots i_k\}$ should be able to incriminate a user by constructing a subset of the coalition keys which is equal to the user's subset U_i. Every k-resilient scheme must have this property (in fact, it has to have a stronger property, that the intersection between the user's subset and the union of the coalition subsets is, at most, $1/k$ the size of a subset). The requirement implies that for all $k+1$ different indices $i_0, i_1, i_2, \cdots i_k$ it should hold that $U_{i_0} \not\subset \cup_{j=1}^{k} U_{i_j}$. In other words, there is a system of n subsets of a universe S with r elements. Each subset contains m elements. These subsets have the property that none of them is contained in the union of k different subsets. Set systems with this "k union property" were investigated by Erdös, Frankl, and Füredi [13]. From [13, Theorem 3.3 and Proposition 3.4], it follows that r is at least $\Omega(\min\{n, k^2\log n/\log\log n\})$. From [13, Proposition 2.1], it follows that $m \ge k\log n/\log r$. These lower bounds imply the following theorem.

Theorem 8: In any open k-resilient traceability scheme, providing every one of the n users with m keys out of r, in a manner which satisfies the "k union property," it holds that

$$r = \Omega(\min\{n, k^2\log n/\log\log n\})$$

and $m \ge k\log n/\log r$.

The lower bounds on both r and m are roughly a factor of k smaller than the best construction we presented for an open traceability system.

VIII. Conclusion

We presented several schemes for tracing users who leak a set of keys, which are good against coalitions of at most k corrupt users. Fully resilient schemes trace the source of keys of any decoder which can decrypt with better probability than breaking the underlying encryption algorithms. The most efficient fully resilient scheme was presented in Section V-B and has an enabling block of length $O(k\log n/p)$. We also presented threshold schemes which trace the source of keys of decoders whose advantage in decryption, over the probability of just breaking the underlying encryption algorithms, is greater than some lower bound. The threshold scheme which was most efficient in terms of data redundancy overhead was presented in Section VI-A. It has an enabling block which contains only $4k$ basic encryptions, regardless of the number of users (n) or the error probability (p). Therefore, the linear dependency on k allows for resiliency against rather large coalitions.

Appendix
A Secret Two-Level Scheme: A Simpler Version

This appendix contains an alternative proof for the security of the two-level secret scheme. The main difference between this proof and the proof of Section V-B is that users are mapped to $2ek/b$ (and not $2ek/b'$) subschemes. The proof presented here might be simpler since it does not use the extra parameter b'. However, the overhead of the scheme is slightly larger.

The construction uses a random mapping h from the domain $\{1, \cdots, n\}$ to a range of size $2ek/b$. For any fixed set of k traitors it holds that the probability that b or more traitors are mapped together by h is, at most,

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

Setting

$$b = \log(1/p) + 2 + \log(ek/\log(1/p)) = \log\left(\frac{4ek}{(p\log(1/p))}\right)$$

implies that this probability is smaller than $p/2$. Once such a mapping is chosen the construction uses the $(b, p/2)$-resilient construction of Section V-A for each set of preimages $h^{-1}(i)$ for $1 \leq i \leq 2ek/b$.

The detection of traitors is performed as in the two-level scheme of Section V-B. The resiliency of the scheme is based on the fact that if no more than b traitors are mapped together (which happens with probability $p/2$), then the probability of incriminating any user (say user 1), is the probability that the subset of traitors that is mapped to $h(1)$ (and by assumption is of size at most b) succeeds in incriminating him.

The number of keys a user gets in this scheme is simply the number of keys a user gets in the $(b, p/2)$-resilient scheme. The size of the enabling block is $2ek/b$ times the size of the enabling block in the $(b, p/2)$-resilient scheme.

We thus obtain the following theorem:

Theorem 9: There is a (p, k)-resilient secret traceability scheme, where a user's personal key contains $\frac{4}{3}b\log(2n/p)$ decryption keys, and an enabling block consists of $\frac{32}{3}ekb\log(2n/p)$ encryptions, where $b = \log\left(\frac{4ek}{(p\log(1/p))}\right)$.

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 Trans. Inform. Theory*, vol. 38, pp. 509–516, Mar. 1992.

[2] N. Alon and J. Spencer, *The Probabilistic Method*. New York: Wiley, 1992.

[3] R. Anderson and M. Kuhn, "Tamper resistance—A cautionary note," in *Usenix Electronic Commerce Workshop*, Oakland, 1996, pp. 1–11.

[4] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Proc. Advances in Cryptology—Crypto '97*, LNCS 1294, 1997, pp. 513–525.

[5] D. Boneh, R. A. Demillo, and R. J. Lipton, "On the importance of checking computations," in *Proc. Advances in Cryptology—Eurocrypt '97*, 1997, pp. 37–51.

[6] D. Boneh and M. Franklin, "An efficient public key tracing scheme," in *Proc. Advances in Cryptology—Crypto '99*, LNCS 1666, 1999, pp. 338–353.

[7] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital date," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1897–1905, Sept. 1998.

[8] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *Proc. INFOCOM '99*, vol. 2, New York, Mar. 1999, pp. 708–716.

[9] J. L. Carter and M. N. Wegman, "Universal classes of hash functions," *J. Comput. Syst. Sci.*, vol. 18, pp. 143–154, 1979.

[10] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *Proc. Advances in Cryptology—Crypto '94*: Springr-Verlag, 1994, LNCS 839, pp. 257–270.

[11] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Processing*, vol. 6, pp. 1673–1687, Dec. 1997.

[12] C. Dwork, J. Lotspiech, and M. Naor, "Digital signets: Self-enforcing protection of digital information," in *28th Symp. Theory of Computation*, 1996, pp. 489–498.

[13] 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. Math.*, vol. 51, pp. 79–89, 1985.

[14] A. Fiat and M. Naor, "Broadcast encryption," in *Proc. Advances in Cryptology—Crypto '93*, 1994, pp. 480–491.

[15] A. Fiat and T. Tassa, "Dynamic traitor tracing," in *Proc. Advances in Cryptology—Crypto '99*, LNCS 1666, 1999, pp. 388–397.

[16] M. L. Fredman, J. Komlós, and E. Szemerédi, "Storing a sparse table with $O(1)$ worst case access time," *J. Assoc. Comput. Mach.*, vol. 31, pp. 538–544, 1984.

[17] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. Assoc. Comput. Mach.*, vol. 33, pp. 792–807, 1986.

[18] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Advances in Cryptology—Crypto '99*, LNCS 1666, 1999, pp. 388–397.

[19] K. Mehlhorn, *Data Structures and Algorithms: Sorting and Searching*. New York: Springer-Verlag, 1984.

[20] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*. Amsterdam, The Netherlands: North Holland, 1977.

[21] M. Naor and B. Pinkas, "Threshold traitor tracing," in *Proc. Advances in Cryptology—Crypto '98*, LNCS 1462, 1998, pp. 502–517.

[22] B. Pfitzmann, "Trials of traced traitors," in *Workshop on Information Hiding*, LNCS 1174, Cambridge, U.K., 1996, pp. 49–64.

[23] D. R. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes," *SIAM J. Discr. Math*, vol. 11, pp. 41–53, 1998.

[24] J. N. Staddon, "A combinatorial study of communication, storage and traceability in broadcast encryption systems," Ph.D. dissertation, Univ. Calif. Berkeley, 1997.

[25] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures," RFC 2627, June 1999.

[26] M. N. Wegman and J. L. Carter, "New hash functions and their use in authentication and set equality," *J. Comput. Syst. Sci.*, vol. 22, pp. 265–279, 1981.