Efficient Trace and Revoke Schemes
(Preliminary Version)

Moni Naor* Benny Pinkas**

Dept. Computer Science and Applied Math
Weizmann Institute of Science
Rehovot 76100, ISRAEL
{naor,bennyp}@wisdom.weizmann.ac.il

Abstract. Our goal is to design encryption schemes for mass distribution of data in which it is possible to (1) deter users from leaking their personal keys, (2) trace which users leaked keys to construct an illegal decryption device, and (3) revoke these keys as to render the device dysfunctional.

We start by designing an efficient revocation scheme, based on secret sharing. It can remove up to t parties and is secure against coalitions of size t. The performance of this scheme is more efficient than that of previous schemes with the same properties. We then show how to combine the revocation scheme with traitor tracing and self enforcement schemes. More precisely, how to construct schemes such that (1) Each user's personal key contains some sensitive information of that user (e.g., the user's credit card number), and therefore users would be reluctant to disclose their keys. (2) An illegal decryption device discloses the identity of users that contributed keys to construct the device. And, (3) it is possible to revoke the keys of corrupt users. For the last point it is important to be able to do so without publicly disclosing the sensitive information.

Keywords: User revocation, blacklisting, broadcast encryption, tracing traitors, self enforcement, copyright protection.

1 Introduction

Digital media is easy to copy. Pirate copying of digital content, such as music, video, or software, is a monumental problem. This copying is done by users who are authorized to use the content but not to redistribute it, and incurs great losses to the producers and distributors of the digital content. This problem affects all forms of digital distribution in media such as music CDs, DVDs, satellite and cable television programs, access to premium database, etc. Our goal is to design schemes that distribute encrypted versions of the content, where it is

* Part of this work was done while visiting Stanford University and IBM Almaden Research Center. Partly supported by DOD Muri grant administered by ONR and DARPA contract F30602-99-1-0530.

** Research supported by an Eshkol Fellowship.

possible to (1) Trace users who leak their decryption keys in order to construct an illegal decryption box, and (2) Revoke those keys so as to render the box dysfunctional. The schemes we propose address both these issues simultaneously, and we call such scheme *trace and revoke schemes*. Furthermore, the schemes have a *self-enforcement* property whereby users are detered from leaking their keys by embedding personal information in them. For ease of presentation we start by describing the revocation problem separately and then go on to deal with traitor tracing and self-enforcement.

User revocation: This work presents simple and efficient methods for *user revocation* (a process also known as user exclusion, or blacklisting). These methods operate in the following scenario: a group of users receives digital content from a *group controller*. The content might be, for example, tv programs or digital music transmitted over satellite or via cables, or stored on DVDs. The content is encrypted, and the decryption key is known to all members of the group. At some point the group controller learns that some users are violating the terms of their usage license[1] (physically, the users might be set-top tv decoders which are known to be used for piracy, or, in DVD systems in which the players have keys for decrypting DVDs, players whose keys were leaked). The group controller must then revoke the decryption capabilities of these users. In a broader scenario, the schemes can be used in a multicast environment for fast rekeying of a multicast group after some parties leave the group.

For a given revocation scheme the important factors that determine its efficiency are (i) The communication overhead, i.e. the length of the messages sent by the center to renew the key. This represents the wasted bandwidth (or in case of DVDs the wasted storage). (ii) Storage overhead by the users, e.g. how many keys they should store. (iii) The computational overhead, especially by the users, to renew the key. The schemes we present enable the revocation of the keys of up to t users from a universe of n users (where t is a parameter), and are secure against coalitions of up to t revoked users. Our schemes are efficient in all three criteria : key length, communication overhead, and computation of the new group key. In particular, *none of these parameters depends on the total number of users, n.* The personal key length is constant, the communication and computation overhead are only linear in t.

We present a very appealing mode of operation for the revocation schemes. It enables removing up to t users in the worst case, with the overhead specified above, but performs much better when only a few users have to be removed. In particular when c users should be removed (where $c < t$), the communication overhead is just c. After removing c users the scheme is ready to remove up to $t - c$ additional users. The group controller can send additional *maintenance* messages to the users (possibly in periods when the bandwidth of the network is

[1] There are tracing methods for finding which users are responsible for distributing illegal copies of the content. See [7,23,4,10], and the discussion below.

not fully utilized) to regain the original worst-case guarantee of the scheme and prepare for revoking up to t new users[2].

Note: a similar revocation scheme to the one we present in Section 2 was discovered independently by Anzai et al [1] (but without the tracing or self-enforcement extensions).

Tracing and self enforcement: While revocation may be applied in several scenarios (e.g. to enforce payments) in this work we emphasize its work in conjunction with methods that combat leaking of keys. There are two *non*-exclusive approaches for combating leakage: (i) trace the corrupt users, which are leaked the keys. I.e. given a pirate box find the source; this is known as *traitor tracing*. (ii) Deter users from revealing their personal keys to others, a task we denote as *self enforcement*.

The self enforcement property is obtained by giving each user a personal key which contains some sensitive information private to him/her, for example the user's credit card number. This personal key is required for the decryption of the content. It is reasonable to assume that users would be reluctant to disclose such personal and sensitive keys to pirates, and in fact few users would even be willing to give these keys to their friends and neighbors.

Self enforcement schemes achieve two goals: (1) They prevent small scale piracy (e.g. a user giving his key to a friend), a task *not* managed by other copyright protection schemes. (2) They make it harder for pirates to obtain users' keys. This goal is very important since most of the complementing schemes that fight piracy (such as our revocation schemes) are successful only if the pirate obtains less than a threshold of t keys, where t is a parameter which affects the complexity of the scheme. Using a self enforcement scheme allows to set lower values for the threshold t, thus improving the efficiency of the schemes.

We describe in Section 3 schemes which enable self enforcement, traitor tracing, and user revocation. The combination of these properties is not limited to revoking users which are found to be corrupt. Section 3.2 describes how to perform *periodic refresh of the group key*, such that only users who have a personal key (which contains their sensitive information) can compute the group key and continue to use the system. This is a very strong security property which is important even for scenarios where it is not expected that users be revoked on a regular basis.

In addition, we describe how to combine revocation with combinatorial tracing schemes, such as those in [7,23].

1.1 Overview of the Results

The scenario: We consider the following scenario. There is a group of n users that share the same key (i.e., the key with which the content is encrypted). A group controller GC is responsible for controlling the decryption capabilities of

[2] This maintenance mode is more appropriate for connected devices such as PCs and set-top boxes, than to an "off-line" device such as DVD.

these users. (The GC might have common key with each of the users, which enables them to communicate via a private channel, but these channels are not directly used by our schemes). The GC prepares keys for the revocation scheme in an initialization phase, and gives each user a personal key. At a certain point a subgroup of up to t users is disallowed from continuing to decrypt the content and therefore a new key should be generated by the GC and become known to all other $n - t$ users. Further group communication should be encrypted with the new key.

Revocation can be trivially achieved as follows: the GC generates a new group key and sends it independently to each of the $n - t$ remaining members of the group, using a private channel between them. This scheme is, however, very inefficient. Its communication overhead is $O(n - t)$ and might be very large, for example in the case of a group a million users, from which a hundred users should be removed. The overhead of our schemes, in contrast, depends only on t.

The basic idea: The basic idea of the revocation scheme is to use secret sharing in the following way: The group controller prepares in advance a key to be used after the revocation. In the initialization phase each user receives a share of this key. In the revocation phase the GC *broadcasts the shares of the revoked users.* Each other user can combine this information with his or her share and obtain the new key, while even a coalition of all the revoked users does not have enough shares to compute any information about this key.

The schemes: We present three types of revocation schemes which can be used to revoke the keys of up to t users, where t is a parameter. The overhead of *all* schemes is the following: each user has a key of constant length, essentially an element in a field (though the field size is different), the revocation message is of length t, and the overhead of the computation of a new key by the user depends only on t.

- Schemes for a single revocation. These are information theoretic secure and can be used for one revocation of up to t users.
- Schemes for many revocations. These can be used to perform many revocations of up to t users in each revocation. They are based on a number theoretic assumption – The Decisional Diffie-Hellman assumption [2]. Note that such schemes are important if the keys are to be changed periodically.
- Self enforcement schemes. We present two schemes: a scheme for many revocation/key changes, which relies on the Decisional Diffie-Hellman assumption, and a scheme for a single revocation which uses pseudo-random functions.

In addition, we present three preferred modes of operation:

- A usage mode for the single revocation schemes, which enables better efficiency if the common operation is the revocation of just a single or a few users.

– Using the self enforcement scheme for *periodic key refreshment*: Once every predefined period of time, the group key is changed using the self enforcement scheme. This ensures that every user which is capable of decrypting the content has a personal key which contains sensitive information.
– Combining revocation with combinatorial tracing schemes.

The most interesting aspect of our schemes is the combination of all three features - revocation, traitor tracing and self enforcement.

1.2 Related Work

Revocation *Broadcast encryption* schemes (Fiat and Naor [15]) enable encrypting messages to an *arbitrary* and dynamically changing subset of the users. Therefore they address a more general problem than revocation schemes[3], (which only allow removal of a limited number of users from the group). When applied to the removal scenario the broadcast encryption schemes can remove *any* number of users under the assumption that at most k of them collude. Broadcast encryption schemes are, therefore, asymptotically more efficient than revocation schemes if the number of users that have to leave the group is large. In particular, the size of a personal key in the most efficient broadcast scheme is logarithmic in k, and the communication overhead is proportional to $k \log^2 k$ and independent of the number of removed elements (the users do need to know the identity of the revoked users, but this is independent of the revocation message).

The *goals* of the work of Kumar et al [18] are similar to those of our basic revocation scheme (Section 2.1). Their method enables a one-time revocation of up to t users, secure against a coalition of all the revoked users. The scheme is based on cover-free sets and they have constructions where the revocation message is of length $O(t \log n)$ as well as $O(t^2)$.

The tree based revocation scheme of [27,28] uses a basic procedure which revokes the key of a single user and updates the keys of all other users in the group. This procedure can be used repeatedly to remove any number of users from the group, and is secure against any coalition of corrupt users. Each user has to keep a key of length $\log n$, and the revocation of each user requires a broadcast message of length $2 \log n$ (the length of this message is reduced to $\log n$ in [5]). The lower bound of [6] demonstrates that these schemes are optimal in some sense. A major problem of revocation schemes of this type is that they require that users receive and process all previous revocation messages in order to be able to update the group key. In particular, a user which rejoins the group after being offline for a while must process all the revocation messages that were sent in its absence. In case these scheme are adapted for the scenario considered in this paper, then to revoke t users implies sending a message containing $O(t \log n)$ keys and this is the computational overhead as well. Key size for the users is $O(\log n)$ encryption keys.

[3] Or at least a different parameterization.

Tracing and self enforcement The goal of *traitor tracing* is to trace the source of keys of illegal decryption devices. Traitor tracing schemes distribute decryption keys to users in a way which guarantees that a pirate decryption device which is constructed using the keys of at most t users (traitors) reveals the identity of at least one of them. The schemes of [7,23] are based on combinatorial and probabilistic constructions, and ensure tracing with high probability. They enable "black box tracing", i.e., tracing when there is no way to examine the inner contents of the pirate decryption device, but it is rather only possible to examine the reply of the device to different ciphertexts.

The *public key tracing* scheme of Boneh and Franklin [3] is based on a number theoretic assumption (Decisional Diffie-Hellman), and has a deterministic tracing guarantee given extracted keys in canonical form. It also has a black-box confirmation test (see discussion at Section 3). In addition, it supports public key encryption in the sense that any party can encrypt messages to the group. We use some of the ideas of [3] for our multi-revocation tracing traitors scheme. A one-time and a multi-time tracing schemes, based on polynomials, are described in [19]. The multi-time tracing scheme was shown to be insecure in [3]. The flaw in the design of that scheme was that although it enabled to trace the source of any single key, it did not prevent the traitors from generating an untraceable combination of their keys, which can serve as a decryption key.

The notion of *self enforcement* was suggested by Dwork, Lotspiech and Naor [12] who also proposed a `signet` scheme with this property. The signets scheme is rather efficient – the computational overhead (for the users) of changing the group key involves a constant number of modular exponentiations and does not depend on the group size or on the size of the coalitions against which the system is secure.

The schemes we develop in this paper can be seen as a combination of the signet scheme [12] and public-key tracing [3].

Combinatorial tracing constructions are further discussed in [26,16] (these use a basic set of independent keys, and assign each user's personal key to be a subset of the set of keys). In particular, the work of [16] discusses the combination of such scheme with revocation schemes[4]. The paper studies two methods for integrating tracing and revocation schemes: adding revocation capabilities to any tracing scheme, and adding tracing capabilities to any revocation scheme.

2 Revocation Schemes

Secret sharing: We base our work on `threshold secret sharing` [25]. A k-out-of-n secret sharing scheme divides a secret into n shares such that no $k-1$ of them disclose any information about the secret but any k shares suffice to recover it. In principal we could apply any secret sharing scheme which maintains a sharp threshold, however, we use Shamir's polynomial based secret sharing scheme [25] which operates as follows. Let $\mathcal{F}$ be a field, and let $S \in \mathcal{F}$ be the secret to be

[4] The terminology of [16] denotes by "broadcast encryption" the schemes that we denote as revocation schemes.

distributed. In order to distribute the secret, a random polynomial of degree $k-1$ is generated over $\mathcal{F}$ subject to the constraint $P(0) = S$. The ith share is defined as $P(i)$. Given any k shares it is easy to interpolate the polynomial and reveal $S = P(0)$ (this requires $O(k \log^2 k)$ multiplications using FFT, or $O(k^2)$ multiplications using Lagrange's interpolation formula). It is straightforward to verify that any $k-1$ shares do not disclose any information about the secret.

2.1 A Scheme for a Single Revocation

The following scheme can be used for a *single revocation of up to t users* with a communication overhead of $O(t)$, and security against a coalition of the t revoked users. We describe in Section 2.1 a usage mode in which a group controller can use a sequence of these schemes to remove any number of users, one after the other, such that if successive revocations do not happen very often then the on-line communication overhead is only $O(1)$ per revocation, and the computation overhead is only $O(t)$.

The basic scheme The scheme operates over a field $\mathcal{F}$ such that a random element in $\mathcal{F}$ can be used as an encryption key of a symmetric (conventional) scheme (e.g. $|\mathcal{F}| > 2^{80}$). Each user u receives an identifier $I_u \in \mathcal{F}$.

Initialization: The GC generates a random polynomial P of degree t over $\mathcal{F}$, (this polynomial can be used for $(t+1)$-out-of-n secret sharing). It sets the secret key S to be used after the revocation to be $S = P(0)$. It provides each user u (over a private channel) with a personal key $K_u = \langle I_u, P(I_u) \rangle$.

Revocation: The group controller learns the identities of t users $I_{u_1}, \ldots, I_{u_t}$ whose keys should be revoked. The GC broadcasts the identities and the personal keys of these users:

$$\langle I_{u_1}, P(I_{u_1}) \rangle, \ldots, \langle I_{u_t}, P(I_{u_t}) \rangle$$

Each other user u can combine his or her personal key K_u with these t keys, and using these $t+1$ shares interpolate P and compute the key $S = P(0)$. The GC uses S as the new group key with which it encrypts messages to the non-revoked users.

If the GC prepares a scheme to revoke t users, and only $t' < t$ users should be removed, it can perform the revocation by sending the shares of these t' users and additional $t - t'$ values of P, at locations which do not equal the identity I_u of any other user.

Theorem 1. *In the above scheme a coalition of all the t revoked users does not have any information about the new key.*

Proof: The property follows immediately from the security of Shamir's secret sharing scheme. $\bowtie$

Note that the GC can add new users to the group even if they join the group after the initialization stage (without any special arrangements). It simply assigns them an identity and sends them the corresponding value of P.

Storage and communication overhead: The secret key that each user has to keep is just a single element of $\mathcal{F}$, i.e. of the same length as the keys that are used to encrypt the communication (the identity I_u need not be secret). The revocation message is of length $2t|\mathcal{F}|$. To further reduce the communication overhead, the identities of the users can be defined in a small subset of $\mathcal{F}$, resulting in a revocation message of length $t(|\mathcal{F}| + \log n)$.

Reducing the computation overhead: The computation of the new group key by a user involves an interpolation of the free coefficient of P, and requires $O(t \log^2 t)$ multiplications using FFT, or $O(t^2)$ multiplications using Lagrange interpolation. This computation overhead can be reduced in two ways: (1) The Lagrange interpolation formula, given $P(u_0), \ldots, P(u_t)$ is $P(0) = \sum_{i=0}^{t} \Pi_{j \neq i} \frac{u_j}{u_j - u_i} P(u_i)$. The GC knows t of the u_i's and can therefore precompute and broadcast, for t Lagrange coefficients, the multiplications between these values. This reduces the computation that a user has to perform in order to compute $P(0)$ to $O(t)$ multiplications, while the communication overhead is only increased by a factor of 2. (2) Instead of using a single polynomial over a field $\mathcal{F}$ of 80 bits, the scheme can use c independent polynomials over a field of size $\mathcal{F}/c$ bits, and use the concatenation of their values at 0 as the new group key. The computation of the new group key involves multiplications over the smaller field, and is more efficient.

Preferred usage mode In a typical scenario the GC should be ready to simultaneously revoke up to t users in the *worst case*, but most of the times it is required to revoke only a single user or a few users. In such cases the revocation scheme can be used according to a mode which enables revocation of up to t users, and can revoke fewer users even more efficiently. In particular, a single user can be revoked with only $O(1)$ communication and $O(t)$ computation (between the time that the need for revocation of the user arises and the actual revocation). After the revocation, the GC sends short maintenance messages to the users, to return the scheme to its original state (i.e., being capable of removing up to t users in the worst case).

Initialization: In the initialization phase the GC prepares t revocation schemes $RS_1, \ldots, RS_t$, such that scheme RS_i can be used to remove i users. That is, scheme RS_i uses a polynomial P_i of degree i. Each user u is given shares from each of the schemes, i.e., u is given a key of length $t+1$, $\langle I_u, P_1(I_u), \ldots, P_t(I_u) \rangle$.

The schemes are used one after the other. Scheme RS_i is used to remove the i'th user that should be revoked (and still prevent the previous $i-1$ revoked users from learning the new key).

First revocation: Suppose that the first user to be revoked is u_1. The GC broadcasts $\langle I_{u_1}, P(I_{u_1}) \rangle$, and all other users use scheme RS_1 to compute $P_1(0)$ which is the new group key. The overhead of this revocation is just $O(1)$, both in communication and in computation.

Maintenance: After removing user u_1, the GC can restore the system to its original state, ready for the revocation of up to t users. It broadcasts the shares of the other polynomials that were known to u_1, namely $P_2(u_1), \ldots, P_t(u_1)$. This broadcast is not urgent (since u_1 is already revoked) and can be done in times in which the network has idle bandwidth. After this broadcast every polynomial P_i, $2 \leq i \leq t$, has only $i-1$ missing shares. For the purpose of secret sharing this reduces the degree of the polynomial by 1, and we can, therefore, denote these polynomials as $P'_1, \ldots, P'_{t-1}$, of degrees $1, \ldots, t-1$ respectively. At this time the GC can revoke up to $t-1$ additional users. To be able to revoke t users, it prepares a new polynomial P'_t of degree t, such that $P'_t(0)$ would serve as the new key after t more users are removed. The GC uses private channels with the users to send shares of P'_t to all users who are currently active (non-revoked).

Note that additional revocations can be performed during the maintenance phase (i.e., before all the shares of P'_t are sent), as long as at most $t-1$ additional users have to be removed. At the end of the maintenance phase the system returns to the state it had before the first revocation and can be used to instantly revoke up to t users. This combination of instant revocation, and system maintenance during off-peak usage, seems optimal for systems which need to prepare for the worst case, but expect only a few revocations during normal operation.

Additional Revocations: The first revocation used a linear polynomial P_1, and, therefore, $P_1(0)$ is computed in constant time. Future revocations of single users employ polynomials of higher degrees, up to degree t. In the case of revocations with a polynomial of degree t, $t-1$ of its shares were broadcast in maintenance phases. Denote such a polynomial as P^*. In the revocation itself users should compute $P^*(0)$. Each user obtained t of the shares in advance (his own share plus the $t-1$ that were broadcast). The user can start the computation of $P^*(0)$ before the last share is broadcast (i.e., before the revocation), and therefore the online overhead of computing $P^*(0)$ is only $O(t)$ (using Lagrange's interpolation), while the communication overhead is only $O(1)$.

Security: The scheme can be used to revoke the keys of an unlimited number of users, as long as at most t of them collude before their revocation.

Overhead: Each user keeps a secret key of length t, which contains its shares for each of the polynomials. In addition, it might keep $O(t^2)$ shares that were broadcast in maintenance phases. The center keeps a secret key of length t^2, i.e., t secret polynomials.

The online communication overhead of revocation is $O(1)$ (a single share). The computation overhead of a revocation of a single user is $O(t)$, and the computation overhead of a revocation of $t' \leq t$ users is $O(tt')$. Each maintenance stage involves the GC sending a single share to each of the users in the group. This overhead is less important since these messages can be sent when the network is idle.

2.2 A Scheme for Many Revocations

The basic polynomial based scheme is good for a single revocation, and the group controller must distribute additional keys (polynomial shares) to support more revocations. We now present a scheme in which each user has a single key which is good for a virtually unlimited number of revocations, as long as at most t revoked users collude together to compute keys they should not receive.

The Decisional Diffie-Hellman assumption The Decisional Diffie-Hellman assumption (DDH) is a most useful cryptographic one. It enables to construct efficient cryptographic primitives with very strong security guarantees. These include the Diffie-Hellman key agreement protocol [11], the El Gamal encryption scheme [13], pseudo-random functions [24], a construction of a cryptosystem secure against chosen ciphertext attacks [8], and more.

The DDH assumption is about a cyclic group G and a generator g. Loosely speaking, it states that no efficient algorithm can distinguish between the two distributions $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^c\rangle$, where a, b, c are randomly chosen in $[1, |G|]$. We refer the reader to [2,24] for further discussions of the assumption.

The scheme The scheme operates over a group Z_q of prime order. More specifically, G_q can be a subgroup of order q in Z_p^*, where p is prime and $q|p-1$. Let g be a generator of Z_q, such that the Decisional Diffie-Hellman assumption holds for Z_q and g. The scheme applies an idea first suggested by Feldman [14] of performing Shamir's secret sharing in the exponents.

Initialization: This process is performed once, for all future revocations. The GC generates a random polynomial P of degree t over Z_q. It sends to user u (via a private channel) a personal key $K_u = \langle I_u, P(I_u)\rangle$ (where I_u is a non-secret identifier associated with u).

Revocation: The GC learns the identities of t users $I_{u_1}, \ldots, I_{u_t}$ that should be revoked. It then chooses a random $r \in Z_q$ and sets $g^{rP(0)}$ to be the new key that would be unknown to the removed users. The GC broadcasts the following message:

$$g^r, \langle I_{u_1}, g^{rP(I_{u_1})}\rangle, \ldots, \langle I_{u_t}, g^{rP(I_{u_t})}\rangle$$

Each non-revoked user u can compute $(g^r)^{P(I_u)}$ and combine it with the broadcasted values, to interpolate the key $g^{rP(0)}$. This is done as follows: Recall Lagrange's interpolation formula for a polynomial P of degree t from its $t+1$ values at points $x_0, \ldots, x_t$,

$$P(0) = \Sigma_{i=0}^{t} \lambda_i P(x_i)$$

where the λ_i's are Lagrange coefficients which depend on the x_i's, i.e. $\lambda_i = \Pi_{j \neq i} \frac{x_i}{x_i - x_j}$. Therefore

$$g^{rP(0)} = g^{r \Sigma_{i=0}^{t} \lambda_i P(x_i)} = \Pi_{i=0}^{t} g^{r \lambda_i P(x_i)}$$

and knowing $t+1$ pairs $\langle I_u, g^{rP(I_u)}\rangle$ allows computing $g^{rP(0)}$.

Theorem 2. *The above scheme can be used for revocations of up to t users in each revocation, and is secure against coalitions of at most t revoked users.*

Proof Sketch: The proof is based on the Decisional Diffie-Hellman assumption. For the sake of clarity we present the details for the case of $t = 1$.

Assume that the scheme with parameter $t = 1$ is insecure and can be broken by user v. This user can be simulated by an algorithm D' which receives the following inputs: a value $P(I_v)$ of the linear polynomial P, polynomially many tuples $\langle g^{r_i}, g^{r_i P(I_v)}, g^{r_i P(0)}\rangle$ generated with randomly chosen r_i's, and a pair $g^r, g^{rP(I_v)}$. If the scheme is insecure then D' can distinguish between $g^{rP(0)}$ and a random value.

We construct an algorithm D that uses D' to break the DDH assumption. D is given input g^a, g^b, and a value C which is either g^{ab} or random. D generates inputs to D' (planning to set $P(0) = b$ and $r = a$). It generates a random key $\langle I_v, P(I_v)\rangle$ and gives it to D'. It then generates random r_i's and gives the tuples $\langle g^{r_i}, g^{r_i P(I_v)}, g^{r_i b}\rangle$ to D'. Then it gives the pair (g^a, C) to D', and outputs the same answer that D' outputs. D's success probability in breaking the DDH assumption is the same as D''s probability of breaking the revocation scheme. $\bowtie$

The scheme enables the GC to remove at most t users, and to add users to the group even if their identities become known only after the initialization stage. The GC can also use several applications of the scheme to revoke the keys of more than t users: in each application it distributes a random key, and sets the new group key to be the exclusive-or of these keys. *The scheme is secure as long as at most t revoked users collude.*

Overhead: The secret key that each user keeps is just a *single* element of Z_q. In order to compute the new key a user should perform t exponentiations; note that we can considerably save here by using simultaneous multiple exponentiations (See Chapter 14.6.1 in [22]). The revocation message is of length $O(t)$. More specifically, it contains $t+1$ elements in Z_p^*, and t elements in Z_q. ($|Z_q|$ can be considerably shorter than $|Z_p^*|$. For example, it is common to set $|Z_q| = 160$ and $|Z_p^*| = 1024$.)

Usage The scheme can be used for virtually unlimited number of revocations, of up to t users per revocation, as long as no more than t revoked users combine their keys in order to learn a key they should not know. Note that after revoking a certain user the GC can decide to restore the access permissions of the user. This does not require the GC to give a new key to that user, and more importantly, does not require sending new keys to any other user. The users can use their old keys for processing future revocation messages that the GC sends.

The scheme is appropriate for scenarios in which very fast revocation is required, but it should also be possible to easily retrieve the capabilities of users that were mistakenly revoked. Consider for example a GC who learns that one of a certain group of users leaked keys to pirates. The GC can quickly revoke the permissions of all the users in the group and prevent further leaks of encrypted content. It is then possible to verify which of these users is helping the pirates, and retrieve the permissions of all other users in this group. This process does not require changing the revocation keys of these users or of the users who were not removed (in fact, they can remain oblivious to the fact that revoked users rejoined the group).

Another case where the properties of the scheme are useful is when the group controller wishes to degrade the quality of the keys of some users (say the keys of users who are late in payments). This will be done by revoking them temporarily out of some content, where the censored information is chosen at random. In more detail, assume that there is a list of ℓ users $u_1, \ldots, u_\ell$ that are late in their payments. To encourage these users to pay their debts, the group controller chooses, once every short period of time, a random subset of t of these ℓ users, and uses the above scheme to distribute a group key which these users cannot decrypt. This key is used to encrypt the content during the next time period. In the next time period the users will be able to decrypt correctly, without additional communication with GC.

3 Schemes that Combine Revocation with Self Enforcement and Tracing

We present two user revocation constructions that have self enforcement *and* tracing capabilities. The first construction is for many revocations or encryptions, and builds upon the signets construction of [12] and the public key construction of [3]. The second construction is for a single revocation, and uses general pseudo-random functions. A delicate issue in self enforcement is that the schemes must preserve the privacy of the removed users. I.e. in the revocation message the GC must not reveal the sensitive information of these users. It should not be easier for someone who did not receive the leaked keys to deduce the sensitive information then without the revocation message (those that received the leaked keys can obtain the sensitive information).

In order to get self enforcement the GC should incorporate in each user's personal key some private information, for example the user's credit card number[5]. Few users would be willing to hand this information to others, and in particular not to pirates who are doing illegal activities. The *tracing* property enables identifying, given an illegal decryption device, which users' keys were used in constructing the device. The combination of these two properties provides a very powerful tool against piracy.

There is, of course, a trivial method for incorporating each user's sensitive information in his or her personal key: The personal key can simply be the sensitive information concatenated to some random data, so that keys of different

[5] In this example it is clear that the center is not allowed to publicly reveal the private information *even if the user has abused the system.*

users are essentially independent. This approach requires the GC to encrypt messages independently to each user, and results in an $O(n)$ communication overhead for a key change in a group of n parties. The schemes that we describe perform much better, and the communication overhead per key change does not depend on the number of users in the group.

The scenario: When a user u registers with the group controller, it provides some private information, S_u. This can be, for example, u's credit card number, which becomes known to the GC as part of the payment process for the content that u is purchasing. The GC then gives u a personal key K_u which operates in conjunction with S_u. Loosely speaking, the system is designed to be secure against coalitions of users in the following sense: using the personal keys of all the coalition members $u_1, \ldots, u_{t'}$, it is impossible to construct a key which does not disclose one of $S_{u_1}, \ldots, S_{u_{t'}}$ and has the same functionality as one of the personal keys.

Regarding the tracing properties of our scheme, there are several possibilities. Given a pirate box and a suspected subset of users we present an effective method for testing whether the box was constructed with the help of the suspected users. This is called *black-box confirmation*. Another possibility is that we manage to extract the key the box uses, in which case, if it is from some *canonical* form, we can find all the contributors to the key. All this can be done assuming there are less than t corrupt users.

3.1 A scheme for many revocations

A natural approach for embedding the user's sensitive information in a scheme like that of Section 2.2 is to make the user identity I_u equal to his sensitive information. However, the problem is that the revocation message includes I_u in the clear, thus revealing the sensitive information of the revoked user to everyone. Instead the key of each user is a pair $\langle x_u, P(x_u)\rangle$ such that $P(x_u)$ enables the extraction of the sensitive information of the user, S_u. This allows sending revocation messages that contain x_u but do not without disclose the sensitive information of the revoked users. We now describe the scheme, first with a simplification of the key assignment.

The scheme The scheme operates as follows:

- **Secret key of the group controller:** a polynomial $P(x) = \sum_{i=0}^{t} a_i x^i$.
- **Key of user u:** The user has sensitive information S_u. It receives a key which is a pair $(x_u, P(x_u))$, s.t. $P(x_u) = S_u$. (Warning - this is refined below).
- **Replacing the group key:** When the GC wishes to replace the key, it chooses a random value r, and sets the new key to be $g^{rP(0)}$. The GC broadcasts a *key change message* which contains g^r and t pairs $(i, g^{rP(i)})$. The user computes $(g^r)^{P(x_u)}$ and interpolates $g^{rP(0)}$.
- **Revocation:** It is possible to revoke up to t users $v_1, \ldots, v_t$. The GC replaces the group key, but instead of broadcasting pairs $(i, g^{rP(i)})$, it broadcasts the t pairs $(x_{v_i}, g^{rP(x_{v_i})})$, using the personal keys of the revoked users.

Generating the user's keys: In order to generate the personal key of user u, the GC should solve the equation $P(x_u) = S_u$. This can be done efficiently using the algorithm of Berlekamp for polynomial factoring in finite fields [9]. There are however several problems with this approach that require refining it:

- There is a chance that this equation has no solution[6].
- While random polynomials of degree t are t-wise independent, we do not know how to show that this t-wise independence is preserved when the query is "in reverse", on the result of the polynomial, as is the case in this scheme[7].
- While not broadcasting the sensitive information S_u does not allow someone with no information on S_u to retrieve it, it does allow *verification* of S_u following revocation by any user who was not revoked.

Instead the solution we propose is to provide the user with a random x_u and $y_u = P(x_u)$. However, in addition publish a public file where S_u is encrypted using y_u. More precisely choose a random s, publish g^s and for each user u encrypt S_u with g^{sy_u}. Also add information that allows searching for this value given g^{sy_u} (e.g. by using a prefix from this string). Therefore any user who leaks y_u is immediately supplying the pirate with a way to obtain S_u.

Analysis

Overhead of revocation: The overhead of the revocation is as in the scheme of Section 2.2.

Properties: The scheme has the following properties:

- *Revocation:* It is possible to revoke up to t users, and the revocation is secure against a coalition of all the t revoked users.
- *Self enforcement:* By disclosing his or her personal key, a user u discloses his or her sensitive information S_u.
- *Tracing:* Any pirate decryption device which was constructed using the keys of up to t users allows confirmation of their identities. Given the actual key, the identities (and sensitive information) can be exposed (here the threshold is $t/2$).

Canonical form of keys: We can view the key a user receives as a vector $K_u = (1, x_u, x_u^2, \ldots, x_u^t)$ and y_u is its inner product with $\boldsymbol{a} = (a_0, a_1, \ldots a_t)$, the coefficients of the polynomial P. Any such key allows reconstruction of a new group key. These are not the only useful keys a small coalition can generate. Consider a coalition $\{u_1, u_2, \ldots u_m\}$ - then for any $\boldsymbol{b} = (b_0, b_1, \ldots b_t)$ which is a linear

[6] This happens with the same probability that a random polynomial of degree t is irreducible, which is roughly $1/t$.

[7] One possible remedy to both problems is to use two polynomials P_1, and P_2 and two user keys x_u^1 and x_u^2 such that $P_1(x_u^1) + P_2(x_u^2) = S_u$. This solution however is not self enforcing since a user can sell "half" a key, i.e. only one of the x_u's.

combination of the vectors $\{K_{u_i}\}$ it is possible for the pirates to compute the inner product of $\boldsymbol{b}$ and $\boldsymbol{a}$. Furthermore, such a vector allows reconstructing the new key following a revocation message (assuming not all members were revoked). However, this *seems*[8] as the only viable option the pirates can take if they want to generate keys allowing reconstruction. We call a vector $\boldsymbol{b}$ and its inner product with $\boldsymbol{a}$, a key in canonical form. What we can show, based on the t-wise independence of P and the DDH is

Lemma 3. *Consider an adversary that receives the keys of a set A of users. Then for any key of canonical form it creates ($\mathbf{b}$ and its inner product with $\mathbf{a}$) it is the case that $\mathbf{b}$ is a linear combination of the vectors of A (or the DDH assumption is false).*

Security of tracing: Once the GC obtains an illegal decryption device, it would like to trace the users that leaked their keys to the device. The tracing properties and methods for our method are similar to those suggested in [3]. We can show that

1. The scheme has a black-box confirmation test, i.e. given a pirate decryption device, and a suspected subset of users, one can test whether all its members were used to generate the device.
2. Given a canonical form key of a pirate device it is possible to extract the subset of users that supplied their values to the key.

In a black-box confirmation test we are given a pirate decryption device as well as candidate subset T of (ab)users. The test outputs “Yes” or “No.” Suppose that the box was really constructed by a pirate group A. Then the output of the test should obey:

- If $T \not\subseteq A$ then “No” whp.
- If $T \supseteq A$ then “Yes” whp.

Note that if $T \subset A$ then the test does not guarantee anything.

Lemma 4. *(Black box confirmation) Given black-box access to a pirate decryption device, and given a subset T of at most t users, it is possible to perform a black-box confirmation test for the subset.*

Proof sketch: The confirmation algorithm generates a polynomial P' which agrees with the keys of T, but not with other keys. It sends a revocation message using P', sets the group key to $P'(0)$, and examines whether the pirate device is able to decrypt. If it decrypts correctly. i.e. has reconstructed $P'(0)$, then it cannot be the case (except with negligible probability) that A does not contain all of T. On the other hand, if the $A \subseteq T$, then the box decrypts correctly with the same probability that it does under normal operation.

[8] We have no proof for that.

Lemma 5. *(Tracing given access to the key of a pirate device) Given a linear combination of at most $t/2$ keys, it is possible to identify the keys that have a non-zero coefficient in the combination.*

Proof sketch: The tracing problem is essentially the following: given a vector that is the linear combination of at most $t/2$ vectors out of the set of all users find this linear combination. This is the decoding problem of a code which is dual to a (Generalized) Reed-Solomon code and is therefore a (Generalized) Reed-Solomon code in itself. (See Chapter 10.8 in [21]). Hence it can be done in polynomial time.

Self enforcement: This extraction procedure also implies self enforcement. A direct application of the self enforcement property only allows to extract a user's sensitive information from his or her personal key, and not from a key which is a linear combination of the personal keys of several users. However, if the pirates use a key in canonical form then they can extract from it personal keys that were used to construct it. These keys reveal the sensitive information of the users that leaked their keys to the pirates, at least to the other members of pirate coalition.
Remark: We do not know how to get full-strength black-box tracing as in [7,23]. I.e. when all the tracer gets to examine is the input/output behavior of the pirate-box and the time it has is much smaller than $\binom{n}{t}$.

3.2 Using the scheme for periodic group key refresh

In a preferred usage mode of the self enforcement revocation scheme, the group controller uses it to change the group key every short period of time (say, once an hour). That is, at the beginning of each period the GC chooses a random value r, sets the group key to be $g^{rP(0)}$, and uses the scheme to let users learn the new key (or, if necessary, to revoke corrupt users).

This usage mode ensures that a party that receives the *group key* from one of the group members can only use it to decrypt the content until the next group key update. It must know a *personal key* in order to compute the new group key and decrypt by itself the content that is being broadcast. Therefore, a legitimate user that wants to enable illegitimate parties to receive the content must either constantly send them the updated values of the group key, or send them a personal key which contains sensitive information.

Public key encryption A variant of the scheme can be used to enable any party to encrypt messages to the group (even if the sender is not a group member), while preserving the revocation, self enforcement, and tracing properties. It is based on the same idea as the scheme of [3].

To enable public key encryption, the GC generates the keys as in Section 3.1. It publishes a public key $\{g^{P(0)}, g^{P(1)}, \ldots, g^{P(t)}\}$ (assuming that no x_u is in the range $[0, t]$). Any party can encrypt a message M by choosing a random r and sending the encryption

$$\langle g^r, g^{rP(0)} \cdot M, g^{rP(1)}, \ldots, g^{rP(t)} \rangle$$

To decrypt, each user u computes $(g^r)^{P(x_u)}$ and uses it to interpolate $g^{rP(0)}$. The security of the encryption is based on the DDH assumption.

To revoke the keys of up to t users $u_1, \ldots, u_t$, the GC chooses a random r' and publishes a new public key:

$$\{g^{r'P(0)}, x_{u_1}, g^{r'P(x_{u_1})}, \ldots, x_{u_t}, g^{r'P(x_{u_t})}\}$$

The encryption and decryption are performed as before.

3.3 A Self Enforcement Scheme for a Single Revocation

We now present a scheme that can be used for a single revocation. Its advantage is computational efficiency: it does not use any number-theoretic assumptions, and, therefore, can use keys which are defined over a relatively small field $\mathcal{F}$ (say, 80 bits long). It uses a combination of a one-way function which hides the users' private information, and a pseudo-random function which prevents users from computing a pseudo-key which does not contain their sensitive information. The idea is to use the basic scheme of Section 2.1, but to effectively release the share the user receives only during revocation.

Initialization: The initialization process is composed of the following steps:

- The GC generates a random polynomial P of degree t, and sets $P(0)$ to be the key to be used after the revocation.
- The GC chooses a random key α, and defines the function $F_\alpha(x, y) = G_\alpha(x) \oplus y$, where G_α is a pseudo-random permutation[9] keyed by α.
- **Generating personal keys:** A user u which wants to use the system provides the GC with his sensitive information S_u. The GC chooses random strings r_u, r'_u, and computes
  - An *identity* for u, $z_u = F_\alpha(S_u, r_u) = G_\alpha(S_u) \oplus r_u$.
  - $P(z_u)$.
  - An encryption of $P(z_u)$: $F_\alpha(r'_u, P(z_u)) = G_\alpha(r'_u) \oplus P(z_u)$.
- The GC sends the values $\langle r_u, r'_u, F_\alpha(r'_u, P(z_u))\rangle$ to the user (note that these do not include z_u or $P(z_u)$). These values together with S_u are u's personal key.

Revocation: To remove users $u_1, \ldots, u_t$, the GC broadcasts the message:

$$\langle \alpha, (z_{u_1}, P(z_{u_1})), \ldots, (z_{u_t}, P(z_{u_t}))\rangle$$

Each user u first uses the key α to compute $z_u = F_\alpha(S_u, r_u) = G_\alpha(S_u) \oplus r_u$, and to decrypt $P(z_u)$. The user then knows $t+1$ values of P and can compute $P(0)$.

[9] A pseudo-random permutation is one that cannot be distinguished from a truly random one by a polynomial-time observer who is given access to the permutation in a black-box manner. (See [17,20] for a precise definition and various constructions). We model a block-cipher as a pseudo-random permutation.

A coalition of all the revoked users does not have any information about the new key, since they have less than $t+1$ values of P. The privacy of the revoked users is guaranteed by the use of a different random string r_u per user, which is not revealed even after the revocation. In fact, even a guess that the private information of u is S_u *cannot* be verified using z_u, because of the use of the random input r_u.

Self-enforcement: The fact that g is a permutation implies that given $G_\alpha(S_u)$ and α we can retrieve S_u. And the pseudo-randomness of g implies that it is impossible to guess $G_\alpha(S_u)$ (and therefore z_u) ahead of time and there *seems*[10] to be no method for combining S_u's which allows to compute z_u once α is released without leaking S_u.

Tracing: as in the previous scheme, we have black-box confirmation using the same technique of choosing a polynomial P' agreeing on T. We don't know how or whether Black-box tracing is possible.

3.4 Combining Revocation with Combinatorial Tracing Schemes

Most tracing schemes (such as those in [7,23]) are based on combinatorial constructions rather than on number theoretic assumptions. In these constructions there is a large set of *independent* basic keys. Each user's personal key is a subset of the basic keys. The schemes encrypt messages in a way which ensures that each personal key enables decryption. On the other hand, the union of the personal keys (i.e., subsets of basic keys) of a coalition of corrupt users (traitors) reveals at least one of the users in that coalition.

Revocation schemes can be combined with tracing schemes in a natural way: A revocation scheme is constructed for each basic key (the basic key corresponds to the group key in the revocation scheme, and the group members are the users whose personal keys include the basic key). Once a user is traced to be a traitor, the basic keys that are included in his or her personal key should be replaced using the corresponding revocation schemes. This would render the pirate decryption device useless (or otherwise, it would be possible to trace another traitor that contributed keys to the device, and revoke his or her keys as well). The storage overhead of this combined scheme is the multiplication of the storage overhead of the tracing scheme by the storage overhead of the revocation scheme. The overhead of removing a traitor is the overhead of a revocation in the revocation scheme, multiplied by the number of basic keys in the personal key. It is appealing to use our revocation schemes in this scenario, since their storage, communication, and computation overhead are low.

Acknowledgments

We thank Russell Impagliazzo for insisting that maintaining the privacy of abusers is important and Ronny Roth and Dan Boneh for helpful advice.

[10] Again, we have no proof for that.

References

1. J. Anzai, N. Matsuzaki and T. Matsumoto, *A Quick Group Key Distribution Scheme with Entity Revocation*. Adv. in Cryptology – Asiacrypt'99, Springer-Verlag LNCS 1716 1999, pp. 333–347.
2. D. Boneh, *The Decision Diffie-Hellman Problem*, in Proceedings of the Third Algorithmic Number Theory Symposium, LNCS Vol. 1423, Springer-Verlag, pp. 48–63, 1998.
3. D. Boneh and M. Franklin, *An efficient public key traitor tracing scheme*, Adv. in Cryptology – Crypto '99, Springr-Verlag LNCS 1666 (1999), 338–353.
4. D. Boneh and J. Shaw, *Collusion-Secure Fingerprinting for Digital date*, Proc. Advances in Cryptology – Crypto '95 (1995), 452–465.
5. 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, pp. 708-716, New York, NY, March 1999.
6. R. Canetti. T. Malkin and K. Nissim, *Efficient Communication-Storage Tradeoffs for Multicast Encryption*, Proc. Advances in Cryptology – Eurocrypt '99, Springr-Verlag LNCS 1592 (1999), 459–474.
7. B. Chor, A. Fiat and M. Naor, *Tracing Traitors*, Proc. Advances in Cryptology – Crypto '94, Springr-Verlag LNCS 839 (1994), 257–270.
8. R. Cramer and V. Shoup, *A practical public key cryptosystem provably secure against adaptove chosen ciphertext attacks*, Proc. Advances in Cryptology – Crypto '98, Springr-Verlag LNCS 1462 (1998), 13–25.
9. H. Cohen, **A course in computational algebraic number theory**, Springer-Verlag, 1996.
10. I. Cox, J. Kilian, T. Leighton and T. Shamoon, *A Secure, Robust Watermark for Multimedia*, Information Hiding Workshop, Cambridge, UK, Springer-Verlag LNCS 1174, (1996), 185–206.
11. Diffie W. and Hellman M. E., New Directions in Cryptography, *IEEE Trans. on Information Theory*, Nov. 1976, 644-654.
12. C. Dwork, J. Lotspiech and M. Naor, *Digital Signets: Self-Enforcing Protection of Digital Information*, 28th Symposium on the Theory of Computation (1996), 489–498.
13. T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, Proc. Advances in Cryptology – Crypto '84, Springr-Verlag LNCS 196 (1985), 10–18.
14. P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, Proc. 28th IEEE Symp. on Foundations of Computer Science, 1987, pp. 427–437.
15. A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology – CRYPTO '93, Springer-Verlag LNCS vol. 773, 1994, pp. 480–491, 1994.
16. E. Gafni, J. Staddon and Y. L. Yin, *Efficient methods for integrating traceability and broadcast encryption*, Proc. Advances in Cryptology – Crypto '99, Springr-Verlag LNCS 1666 (1999), 372–387.
17. O. Goldreich, S. Goldwasser and S. Micali, *How to construct random functions*, *J. of the ACM.*, vol. 33, 1986, pp. 792-807.
18. R. Kumar, S. Rajagopalan and A. Sahai, *Coding constructions for blacklisting problems without computational assumptions*, Adv. in Cryptology – Crypto '99, Springr-Verlag LNCS 1666, pp. 609–623, 1999.
19. K. Kurosawa and Y. Desmedt, *Optimum traitor tracing and asymmetric schemes*, Adv. in Cryptology – Eurocrypt '98, Springr-Verlag LNCS 1403 (1998), 145–157.

20. M. Luby, **Pseudo-randomness and applications**, Princeton University Press, 1996.
21. F. J. MacWilliams and N. J. A. Sloane, **The Theory of Error-Corecting Codes**, North Holland, Amsterdam, 1977.
22. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, **Handbook of Applied Cryptography**, CRC Press, 1996.
23. M. Naor and B. Pinkas, *Threshold Traitor Tracing*, Proc. Advances in Cryptology – Crypto '98, Springr-Verlag LNCS 1462 (1998), 502–517.
24. M. Naor and O. Reingold, *Number-Theoretic constructions of efficient pseudo-random functions*, Proc. 38th IEEE Symp. on Foundations of Computer Science, 1997, pp. 458–467.
25. A. Shamir, How to share a secret, *Comm. ACM*, Vol. 22, No. 11, 1979, 612–613.
26. D. R. Stinson and R. Wei, *Combinatorial properties and constructions of traceability schemes and frameproof codes*, SIAM J. on Discrete Math, Vol. 11, 1, 1998, 41–53.
27. D.M. Wallner, E.J. Harder and R.C. Agee, *Key Management for Multicast: Issues and Architectures*, Internet Request for Comments 2627, June, 1999. Available: `ftp.ietf.org/rfc/rfc2627.txt`
28. C.K. Wong, M. Gouda and S. Lam, *Secure Group Communications Using Key Graphs*, Proc. of ACM Sigcomm '98, Sept. 2-4, Vancouver, Canada, pp. 68–79.