Efficient Trace and Revoke Schemes

Moni Naor* Benny Pinkas†

Abstract

Our goal is to design encryption schemes for mass distribution of data that enable to (1) deter users from leaking their personal keys, (2) trace the identities of users whose keys were used to construct illegal decryption devices, and (3) revoke these keys as to render the devices dysfunctional.

We start by designing an efficient revocation scheme, based on secret sharing. It can remove up to t parties, is secure against coalitions of up to t users, and is more efficient than previous schemes with the same properties. We then show how to enhance the revocation scheme with traitor tracing and self enforcement properties. 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), in order to make users 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, broadcast encryption, tracing traitors, self enforcement, copyright protection.

1 Introduction

Digital media is easy to copy and manipulate. While this has brought many useful applications it has also made pirate copying of digital content, such as music, video, or software, a significant problem. This copying is done by users who are authorized to use the content but are not authorized 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 various types of media such as music, DVDs, satellite and cable television programs, access to premium database, etc. Our goal is to design systems that prevent the abuse of legitimate distribution channels. In particular we design schemes that support distribution of encrypted versions of content, while enabling the following features:

1. Deterring users from disclosing their keys to other parties.
2. Tracing users who leak their decryption keys to pirates in order to construct an illegal decryption box, and

*Dept. of Computer Science and Applied Math, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: naor@wisdom.weizmann.ac.il. Part of this work was done while visiting Stanford University. Partly supported by DARPA contract F30602-99-1-0530.

†STAR Lab, Intertrust Technologies. Most of this work was done while at the Dept. of Computer Science and Applied Math, Weizmann Institute of Science, Israel. E-mail: bpinkas@intertrust.com.

3. Revoking those keys so as to render the pirate decryption box useless.

The schemes we propose address the tracing and revocation issues simultaneously. We call such schemes *trace and revoke schemes*. Furthermore, the schemes have a *self-enforcement* property whereby users are deterred 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 channels such as the Internet, satellite broadcasting, cables, or 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 (for example, the users might be set-top TV decoders that are known to be used for piracy, or, in the case of DVD systems in which the players have keys for decrypting DVDs, players whose keys were leaked). There are tracing methods for finding which users are responsible for distributing illegal copies of the content. See [8, 27, 5, 12], and the discussion below. The group controller must then revoke the decryption capabilities of these users. In a broader scenario, user revocation schemes can be used in a multicast environment for fast rekeying of a multicast group after some parties leave the group [6].

For a given revocation scheme the important factors that determine its efficiency are (1) 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). (2) *Storage* overhead by the users, e.g. how many keys they should store. (3) The *computational* overhead of key update, especially by the users.

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 overheads are only linear in t.

We present a very appealing mode of operation for revocation. It enables to remove 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 not fully utilized) to regain the original worst-case guarantee of the scheme and prepare for a revocation of up to t new users (this maintenance mode is more appropriate for connected devices such as PCs and set-top boxes, than to an "off-line" device such as DVDs).

We note that a revocation scheme similar 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 deployment in conjunction with methods that combat leaking of keys. There are two *non*-exclusive approaches for combating leakage: (1) Tracing the corrupt users which leaked their keys. I.e. given a pirate box finding the source of its keys; this is known as *traitor tracing*. (2) Deterring 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 even that few users would be willing to give these keys to their friends and neighbors.

Self enforcement schemes achieve two goals: They prevent small scale piracy (e.g. a user giving his key to a friend), a task *not* managed by other copyright protection schemes. They also 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 overhead of the scheme. Using a self enforcement scheme justifies the use of a smaller threshold t, thus improving the efficiency of the schemes.

We describe in Section 3 schemes that enable self enforcement, traitor tracing, and user revocation. The combination of these properties is not limited to revoking users that are found to be corrupt. Section 3.6 describes how to perform *periodic refresh of the group key*, such that only users that have a personal key (which contains their sensitive information) can compute an updated group key and continue to use the system, while users that only have the group key and no personal key cannot keep the value of the group key updated. This is a very strong security property that 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 [8, 27].

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 these users. The GC might have a common secret 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 trace and revoke 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 in the following way. 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. (A typical a group might include 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 our 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 its own 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 the new key.

The schemes: We present three types of revocation schemes that 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, the revocation message is of length $O(t)$, and the overhead of computing a new key by a user depends only on t.

- Schemes for a single revocation. These schemes are information theoretic secure and can be used for one revocation of up to t users.
- Schemes for many revocations. These schemes 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 [3]. These schemes are important if the keys are to be changed periodically.
- A Scheme with tracing and self enforcement. We present a scheme for many revocations/key changes, which relies on the Decisional Diffie-Hellman assumption.

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 only a single or a few users.
- Using the self enforcement scheme for *periodic key refreshment*: The group key is changed periodically using the self enforcement scheme. This ensures that every user that is capable of decrypting the content has a personal key that 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 [17]) enable the GC to encrypt messages to an *arbitrary* and *dynamically* changing subset of the users. Therefore they address a more general problem than revocation schemes, or at least a different parameterization (which allows removal of an arbitrary number of users from the group with an overhead that does not depend on the number of removed users). 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 must 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. [21] 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. They present several constructions based on cover-free sets with revocation messages of length $O(t \log n)$ as well as $O(t^2)$.

The tree based revocation scheme of [33, 35] uses a basic procedure that 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 store $\log n$ keys, 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 [6]). The lower bound of [7] demonstrates that these schemes are optimal in some sense. A major problem of revocation schemes of this type is that they require users to receive and process all previous revocation messages in order to be able to update the group key. In particular, a user that rejoins the group after being offline for a while must process all the revocation messages that were sent in his absence. If these schemes are adapted for the scenario considered in this paper, then we get the following performance: a revocation of t users implies sending a message with $O(t \log n)$ keys, as well as a similar computational overhead. The key of each users contains $O(\log n)$ encryption keys.

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 that guarantees that a pirate decryption device that is constructed using the keys of at most t users (traitors) reveals the identity of at least one of them. The schemes of [8, 27, 9] 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, and where it is only possible to examine the reply of the device to different ciphertexts.

The *public key tracing* scheme of Boneh and Franklin [4] is based on a number theoretic assumption (the Decisional Diffie-Hellman assumption), 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. Our multi-revocation traitor tracing scheme uses many of the ideas of [4], in particular the concepts of black-box confirmation and the tracing via decoding. A one-time and a multi-time tracing schemes, based on polynomials, are described in [22]. The multi-time tracing scheme of [22] was shown to be insecure in [31, 4]. The flaw in the design of that scheme is that although it enables to trace the source of any single key, it does not prevent the traitors from generating an untraceable

combination of their keys, which can serve as a decryption key.

In a recent work Kiayias and Yung [20] examined the conditions that enable a tracing scheme to support "black-box tracing", namely tracing the identities of corrupt users while treating the pirate decoding device as a black-box that need not be opened (this is compared to a tracing algorithm that requires to reverse engineer the pirate decoder and reveal the keys that it uses). One of the results in [20] shows that the public key tracing schemes of [4, 22] cannot support black-box tracing if the number of traitors is $\omega(\log n)$, while the schemes of [8, 27, 9] support black-box tracing in these situations. The tracing schemes we suggest fall into the former category and therefore cannot support black-box tracing if the number of traitors is $\omega(\log n)$.

Naor, Naor and Lotspiech [26] describe two very efficient methods supporting both revocation and tracing. The schemes are secure against coalitions of *any* size. They require users to store personal keys of length $\log n$ and $\frac{1}{2}\log^2 n$ keys respectively (where n is the total number of users). Revocation of r users is done using messages of length $r \log N$ and $2r$ keys respectively.

The notion of *self enforcement* was suggested by Dwork, Lotspiech and Naor [14] who also proposed a `signets` 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 signets scheme [14] and public-key tracing [4].

Combinatorial tracing constructions are further discussed in [32, 18]. They 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 [18] discusses the combination of such scheme with revocation schemes (although in the terminology of [18] "broadcast encryption" refers to schemes that we denoted as revocation schemes). It describes two methods for integrating tracing and revocation schemes: adding revocation capabilities to any tracing scheme (at the cost of implementing the revocation using an OR protocol, whose communication overhead is high), and adding tracing capabilities to any revocation scheme (at the cost of increasing the number of keys by a factor of $2t^2$).

2 Revocation Schemes

Secret sharing: We base our work on *threshold secret sharing* [29, 2]. A k-out-of-n threshold secret sharing scheme divides a secret into n shares such that no $k-1$ of them disclose any information about the secret while any k shares suffice to recover it. In principle we could apply any secret sharing scheme that maintains a sharp threshold, however, we choose to use Shamir's polynomial based secret sharing scheme [29] that operates in the following way. Let $\mathcal{F}$ be a field, and let $S \in \mathcal{F}$ be the secret to be shared. In order to share the secret a random polynomial P 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 all the t revoked users.

2.1.1 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 encryption scheme (e.g. $|\mathcal{F}| > 2^{80}$). (This is required since this key is used as the group key after the revocation.) Each user u receives an arbitrary 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 $S = P(0)$. The GC 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 its 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 that are different from 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, since the coalition has only t shares. □

Note that the GC can add new users to the group even if they join the group after the initialization stage. It simply assigns them an identity and provides them with the corresponding value of P.

Storage and communication overhead: The secret key that each user has to keep is 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 $2t|\mathcal{F}|$ (where $|\mathcal{F}|$ is the size of a representation of an element from $\mathcal{F}$). 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:

- The Lagrange interpolation formula is, given $P(I_{u_0}), \ldots, P(I_{u_t})$,

$$P(0) = \sum_{i=0}^{t} \Pi_{j \neq i} \frac{I_{u_j}}{I_{u_j} - I_{u_i}} P(I_{u_i}).$$

  The GC knows the identities of t of the u_i's, namely the revoked users, and can therefore precompute and broadcast, for t Lagrange coefficients, the corresponding multiplications between these values. This reduces the computation that a user has to perform in order to compute $P(0)$ to $O(t)$ multiplications, at the cost of increasing the communication overhead by $|\mathcal{F}|t$ more bits.

- Instead of using a single polynomial (say, 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 therefore more efficient (furthermore, the Lagrange coefficients should be computed only once, since the same coefficients are used for all the polynomials).

2.1.2 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 t copies of the basic revocation scheme can be used to enable revocation of up to t users in the worst case, and enable more efficient revocation of fewer users. 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 a share from each of the schemes, i.e., u is given a key of length $t+1$, consisting of $\langle I_u, P_1(I_u), \ldots, P_t(I_u) \rangle$.

The schemes are used one after the other. Scheme RS_i is used to remove the ith 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 defined to be the new group key. Both the communication and computation overhead of this revocation are $O(1)$.

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 $P_2(I_{u_1}), \ldots, P_t(I_{u_1})$. This broadcast is not urgent (since u_1 is already revoked) and can be done when 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, and $P'_t(0)$ is defined to be the new key after t more users would be 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 revoke up to t users. This combination of instant revocation, and system maintenance during off-peak usage, seems optimal for systems that need prepare for the worst case, but expect only a few revocations during normal operation.

Additional Revocations: The first revocation uses 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. After the tth revocation, all revocations use polynomials of degree t. Consider a revocation with a polynomial of degree t that $t-1$ 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 $t-1$ shares 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: Each polynomial can be used to revoke up to t users. Since the maintenance steps generate new a polynomial whenever a new polynomial is used, the system 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 that contains his shares for each of the polynomials. In addition, he might keep $O(t^2)$ shares that were broadcast in maintenance phases. The center keeps a secret key of length $O(t^2)$, i.e., the coefficients of t secret polynomials of degree $t+1$.

The online communication overhead of a revocation is $O(1)$ (a single share). The online computation overhead of revoking a single user is $O(t)$ (although the overhead of revoking each of the first t users is smaller), and the online 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 should be 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 requires the group controller to distribute additional keys (shares) to support more revocations. In the following scheme each user has a single key that 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 scheme is based on the Decisional Diffie-Hellman assumption.

2.2.1 The Decisional Diffie-Hellman assumption

The Decisional Diffie-Hellman assumption (DDH) is useful for constructing efficient cryptographic primitives with very strong security guarantees. These include the Diffie-Hellman key agreement protocol [13], the El Gamal encryption scheme [15], pseudo-random functions [28], a construction of a cryptosystem secure against chosen ciphertext attacks [10], and more.

The DDH assumption involves 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 [3, 28] for further discussions of the assumption.

2.2.2 Revocation schemes

The schemes operate over a group Z_q of prime order. More specifically, Z_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 schemes applies an idea first suggested by Feldman [16] of doing 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 publishes p and q and 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.

We suggest two variants of revocation. The first can be used for many revocations as long as at most t users are prevented from learning the group key at any given time. The second method can be used for many revocations of an unlimited number of users, as long as less then t of them should be revoked in a single revocation.

Revocation method 1: 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 should be unknown to the removed users. The GC broadcasts the following message (in the clear):

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

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 that depend on the x_i's, i.e. $\lambda_i = \Pi_{j \neq i} \frac{x_j}{x_j - x_i}$. Therefore

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

Given $t+1$ pairs $\langle I_u, g^{rP(I_u)} \rangle$ this formula shows how to compute $g^{rP(0)}$.

Revocation method 2: This method is identical to method 1, except for the GC broadcasting the revocation message $(g^r, \langle I_{u_1}, g^{rP(I_{u_1})} \rangle, \ldots, \langle I_{u_t}, g^{rP(I_{u_t})} \rangle)$ encrypted using the current group key. This ensures that only current group members can read this message.

Theorem 2 *Revocation method 1 can be used for repeated revocations as long as up to t users should be prevented from learning the group key at any given time. The method is secure against coalitions of at most t revoked users. Namely, such a coalition cannot distinguish between a group key it should not learn and a random value.*

Proof: The proof is based on the Decisional Diffie-Hellman assumption. For the sake of clarity we first 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 runs an algorithm D' that receives the following inputs: a value $P(I_v)$ of the linear polynomial P and 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 $g^r, g^{rP(I_v)}$. (The tuples $\langle g^{r_i}, g^{r_i P(I_v)}, g^{r_i P(0)} \rangle$ become known to the user during revocation operations in which other users were revoked. In addition the user might of course learn other values of exponents of the polynomial, but these can be computed from the values in $\langle g^{r_i}, g^{r_i P(I_v)}, g^{r_i P(0)} \rangle$). If the scheme is insecure then D' can then distinguish between $g^{rP(0)}$ and a random value.

Construct an algorithm D that uses D' to break the DDH assumption. D is given input g^a, g^b, and a value C that 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 values $(g^a, g^{aP(I_v)}, C)$ to D', and outputs the same answer that D' outputs for the decision whether C is equal to g^{ab} or not. D's success probability in breaking the DDH assumption is the same as the probability of D' breaking the revocation scheme.

Now consider a coalition of t corrupt users, say users $1, \ldots, t$. These users run an algorithm D' that receives the following inputs: values $P(I_1), \ldots, P(I_t)$ of the linear polynomial P at locations $I_1, \ldots, I_t$, polynomially many tuples $\langle g^{r_i}, g^{r_i P(I_1)}, \ldots, g^{r_i P(I_t)}, g^{r_i P(0)} \rangle$ generated with randomly chosen r_i's (these values became known to the coalition from revocation messages in which at least one of the coalition members was not revoked), and a tuple of the form $g^r, g^{rP(I_1)}, \ldots, g^{rP(I_t)}$ (that is, for every value $P(I_u)$ known to the coalition, there is a corresponding value $g^{rP(I_u)}$ in the tuple). If the scheme is insecure then given this information D' can distinguish between $g^{rP(0)}$ and a random value.

Using D' we can construct an algorithm D that breaks the DDH assumption. It is given input g^a, g^b, and a value C that is either g^{ab} or random. D then generates inputs to D' (planning to set $P(0) = b$ and $r = a$). It generates random keys $\{\langle I_j, P(I_j) \rangle\}_{j=1}^{t}$ and gives them to D'. It then generates random r_i's and gives the tuples $\langle g^{r_i}, g^{r_i P(I_1)}, \ldots, g^{r_i P(I_t)}, g^{r_i b} \rangle$ to D'. Then it gives the values $(g^a, g^{aP(I_1)}, \ldots, g^{aP(I_t)}, C)$ to D', and outputs the same answer that D' outputs regarding whether C is equal to g^{ab} or not. D's success probability

in breaking the DDH assumption is the same as D''s probability of breaking the revocation scheme. □

Theorem 3 *Revocation method 2 can be used for repeated revocations of an unlimited number of users, and supports revoking up to t users at any invocation. It is secure against coalitions of at most t revoked users.*

Proof: Consider any coalition of t revoked users. The revocations of these users could have occurred in several rounds. Denote the group key that is communicated in the ith revocation as S_i. Suppose that all members of the coalition were revoked and consider the round, say round ℓ, in which the last coalition member was revoked (it could be that coalition members move in and out of the group, but we are interested in a time in which none of them is a group member, and in particular $S_{\ell-1}$ is the last group key that any of the coalition members should know). Theorem 2 ensures that the coalition members cannot distinguish between S_ℓ, the group key sent in round ℓ, and a random value. Any future group key, S_m, $m > \ell$, is independent of the information sent *before* the mth revocation. If the coalition can distinguish S_m from a random value then (assuming that the encryption functions that use S_m are secure) it can distinguish between the information sent in the mth revocation and random. Assuming that the encryption function is secure, this means that the coalition can distinguish between S_{m-1}, the group key with which the mth revocation message was encrypted, and random. Repeating this argument $m - \ell$ times we get that the coalition can distinguish between S_ℓ and random. A contradiction. □

Note that the scheme enables the GC to add users to the group even if their identities become known only after the initialization stage.

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 the overhead can be considerably reduced by using simultaneous multiple exponentiations (See Chapter 14.6.1 in [25]). 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 $|Z_p^*| = 1024$.)

2.2.3 Usage

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 all 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 whose keys were mistakenly revoked. Consider for example a GC that 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 leakage of encrypted content. It is then possible to verify which of these users is helping the pirates, and restore 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 revoked (in fact, they can remain oblivious to the fact that revoked users rejoined the group).

Another useful application is where 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 can 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 that these users cannot decrypt. This key is used to encrypt the content during the next time period. In the following time period these users will be able to decrypt correctly without additional communication with the GC.

3 Combining Revocation with Self Enforcement and Tracing

We present a user revocation construction with self enforcement *and* tracing capabilities. The construction is for many revocations, and builds upon the signets construction of [14] and the public key tracing construction of [4]. A delicate issue in self enforcement is that the scheme must preserve the *privacy* of the revoked users. Namely, the revocation message must not reveal the sensitive information of these users. (In other words, although users that give their keys to pirates reveal their sensitive information to the pirates, we do not want the revocation mechanism to reveal this information to other users.)

In order to obtain the self enforcement property the GC should incorporate in each user's personal key some private information, for example the user's credit card number or social security number (in these cases it is clear that the center is not allowed to publicly reveal the private information even if the user has abused the system.). 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 to identify, 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 the personal key: The personal key can simply be the sensitive information concatenated to some random data, so that keys of different 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, in particular 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 GC 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 that operates in conjunction with S_u. Loosely speaking, self enforcement means that any useful key that the user gives to a pirate must contain S_u. Tracing means that using the personal keys of the members of a coalition of t members $u_1, \ldots, u_t$, it is impossible to construct a decoder that does not disclose the identity of one of $u_1, \ldots, u_t$ and has the same functionality as one of the personal keys.

There are different kinds of tracing properties that are supported by our scheme:

- **Black-box confirmation:** Given a pirate box and a suspected subset of at most t users we present an effective method for testing whether the box was constructed with the help of the suspected users, as long as the keys of at most t corrupt users were used to generate the pirate decoding box. This is called black-box confirmation since there is no need to "open" the box and find the explicit key that it uses. It is sufficient to treat the decoder as a black box and examine how it reacts to different messages it receives.

- **Tracing:** Better tracing can be achieved if the tracing process is able to to examine the contents of the decoding device and extract the keys that it uses. In this case, if the keys are in a *canonical* form (defined below), and at most $t/2$ users contributed keys to the pirate, the tracing algorithm can find all the contributors to the key.

A note on black-box tracing: Black-box tracing is of course preferable to a tracing algorithm that requires to "open" the pirate decoding device and identify the key that it uses. We were not able to design a black-box tracing algorithm, except for an $O(\binom{n}{t})$ algorithm that uses black-box confirmation by starting from a group of suspects that contains the set of traitors and then narrowing it down until a traitor is identified. Our inability to support black-box tracing is not surprising given the recent result of Kiayias and Yung [20], which showed that black-box tracing is impossible in a system like ours if the number of traitors is $\omega(\log n)$.

3.1 A Simple 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 or her sensitive information. The problem however is that the revocation message includes I_u in the clear, thus revealing the sensitive information of the revoked user to everyone. Instead we define the key of each user to be a pair $\langle x_u, P(x_u)\rangle$ such that $P(x_u)$ enables the extraction of the sensitive information of the user. This allows sending revocation messages that contain the coordinate x_u in which a user's share is defined, but do not disclose the sensitive information of the revoked users. We describe the scheme, first with a simplification of the key assignment.

The simplified scheme operates over a group Z_q of prime order, for example where Z_q is 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 operates as follows:

- **Secret key of the group controller:** a polynomial $P(x) = \sum_{i=0}^{t} a_i x^i$ in Z_q.

- **Key of user u:** The user has sensitive information S_u. It receives a key that is a pair $(x_u, P(x_u))$, s.t. $P(x_u) = S_u$. (Warning - this step is refined below).

- **Replacing the group key:** When the GC wishes to replace the key (without any user revocation), it chooses a random value r, and sets the new key to be $g^{rP(0)}$. The GC broadcasts a *key change message* that contains g^r and t pairs $(i, g^{rP(i)})$ (The i

values can be arbitrary as long as they do not equal x_u for any user u). Each user computes $(g^r)^{P(x_u)}$ and interpolates $g^{rP(0)}$ using the $t+1$ values of $g^{rP(\cdot)}$ that it knows.

- **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 t pairs $(x_{v_i}, g^{rP(x_{v_i})})$, which are generated using the personal keys of the revoked users.

Generating the users 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 factoring polynomials in finite fields [11]. There are however several problems with this approach that require refining it:

- There is a chance that the equation $P(x_u) = S_u$ has no solution. This happens with the same probability that a random polynomial of degree t is irreducible, which is roughly $1/t$.
- While random polynomials of degree t are $(t+1)$-wise independent, we do not know how to show that this independence is preserved when the query is "in reverse" (i.e. where the result of the polynomial $P(x)$ is given and the point x is then computed, as is the case with this scheme).
- 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 $P_1(x_u^1) + P_2(x_u^2) = S_u$. The new group key will be $g^{rP_1(0)} + g^{rP_2(0)}$. This solution could provide tracing but is not self enforcing since a user can sell "half" a key, i.e. only one of the x_u's.
- The fact that the sensitive information S_u is not being broadcast prevents someone with no information about S_u from retrieving it, but the protocol does not prevent *verification* of an a-priori guess about S_u by any user who was not revoked. (For example, if S_u is the user's mother maiden name and u is revoked, one could check whether this name is one of the 100 most common English names, by checking if g^r to the power of any of these names is equal to $g^{rP(u)}$.)

3.2 The Revised Scheme

In order to avoid the problems listed above the scheme should set the personal keys of users to be the values of P at randomly chosen locations. In other words, the scheme should be identical to the description given in Section 3.1, except for the following exception:

> Each user is provided with a random x_u and $y_u = P(x_u)$. In addition a public file is published, where S_u is encrypted using y_u.

Note that the public file should be available for any interested party, but there is no need to send the file itself to every user (say, broadcast or distribute it together with the content). It is sufficient to provide links to the public file, and make it clear that any party that obtains the personal key of a user can use the public file to obtain the user's private information.

The encryption of the information in the public file should be done using El Gamal encryption method, in the following way. The GC chooses a random s, publishes g^s in the

file, and for each user u encrypts the information S_u using g^{sy_u} as the key. Using the prefix-truncation method of [26], the encryption can be done as $g^s, \{H(g^{sy_u}) \oplus S_u\}$, where H is a pair-wise independent hash function. Note that the length of the encrypted information is only about $|S_u|$ times the number of users, although public-key cryptography is used. This encryption method should be recommended since its security is based on the DDH assumption, as is the security of the revocation scheme. (A naive solution that uses y_u as the key for a symmetric encryption scheme might not be secure, since it requires y_u to be used as the secret key in two different encryption schemes, the revocation scheme and the symmetric scheme. Although y_u can be securely used as a key in one cryptosystem, it is not clear whether a combined use in two cryptosystems affects the overall security. See [30] for an analysis of the use of dependent keys in two cryptosystems.) The public file should also include information that allows searching for a value given the key g^{sy_u}, e.g. by using a prefix from this string as an index. Therefore any user who leaks y_u is immediately supplying the pirate with a way to obtain S_u.

A note on privacy: Any coalition that obtains the secret keys of $t+1$ users can compute any value $g^{sy_u} = g^{sP(x_u)}$ given the value x_u. Therefore such a coalition can learn the sensitive information of revoked users, since their x_u values are published in the revocation message, and their personal information is available, encrypted with g^{sy_u}, in the public file.

3.3 Analysis

Overhead of revocation: The overhead of the revocation is as in the scheme of Section 2.2.2 since the revocation properties are essentially the same. In more detail, the secret key that each user keeps is a single element of Z_q. A user should perform t exponentiations in order to compute a new key, and this overhead can be reduced by using simultaneous multiple exponentiations (Chapter 14.6.1 in [25]). The revocation message is of length $O(t)$, containing $t+1$ elements in Z_p^* and t elements in Z_q.

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. This property follows from Theorem 2.
- *Self enforcement:* By disclosing its personal key, a user u discloses its sensitive information S_u. This follows from the discussion given above.
- *Tracing:* Once the GC obtains an illegal decryption device it would like to trace the users whose keys were used to construct the device. The tracing properties and methods for our method are similar to those suggested in [4]. We show below that
  1. The scheme has a black-box confirmation test, i.e. given a pirate decryption device and a suspected subset of at most t users, one can test whether the members of the subset contributed keys that were used to generate the device.
  2. Given a key of a pirate device that is in canonical form and that was constructed using the keys of at most $t/2$ users it is possible to extract the subset of users whose keys were used to generate the key.

3.4 Black-box confirmation

Consider a scenario in which the tracing system is given a subset of users that are suspected of providing keys for the pirate device. We define a "black-box confirmation test" for the schemes in Sections 3.1,3.2. This test confirms whether these users are indeed traitors.

Definition 4 (black-box confirmation test) *A black-box confirmation test is an algorithm whose input is a pirate decryption device as well as subset C of candidate (ab)users, and whose output is either "Yes" or "No." Suppose that the box was really constructed by a pirate group T of traitors. Then the output of the test should obey:*

- *If $C \cap T = T$ (namely, the real group of traitors is contained in the subset of candidate suspects) then the algorithm should output "Yes" with high probability.*
- *If $C \cap T = \emptyset$ (the subset of suspects does not contain any of the real traitors) then the algorithm should output "No" with high probability.*

Note that if $C \cap T \neq \emptyset$ and $C \cap T \neq T$ then the test does not guarantee any result.

Construction 1 (Black-box confirmation test) *If $|C| < t$ then the confirmation test generates a random set $\hat{C}$ of users, subject to the constraints that $C \subset \hat{C}$ and $\hat{C} = t$. Otherwise it sets $\hat{C} = C$. The confirmation test then generates a random polynomial P' of degree t, subject to the constraint that it agrees with the keys of $\hat{C}$. Namely, for every $u \in \hat{C}$ it holds that $P(u) = P'(u)$ and for any other value v, $P(v)$ is independent of $P'(v)$.*

The confirmation test then picks a set R of t random values $r_1, \ldots, r_t \in \mathcal{F}$, and sends a revocation message using P', revoking the keys of the users whose identities are in R (namely, the revocation message uses the values $g^{rP'(r_1)}, \ldots, g^{rP'(r_t)}$). The group key is set to $g^{rP'(0)}$ and the test examines whether the decryption device is able to decrypt messages encrypted with the new key. If the decryption succeeds the output of the test is "Yes", otherwise the output is "No".

Claim 5 *Given black-box access to a pirate decryption device generated with the keys of at most t users, and given a subset C of at most t users, Construction 1 above is a black-box confirmation test for C.*

Proof: First, note that with high probability the set R of t random values in $\mathcal{F}$ does not contain any element from C or T.

If $C \cap T = T$ then the pirate decryption device cannot distinguish between a regular revocation message and the revocation message sent in the test. The pirate device should therefore be able to decrypt messages using the new group key and the output of the confirmation test is therefore "Yes".

If $C \cap T = \emptyset$ then the new key $g^{rP'(0)}$ is independent of the keys that were used to generate the pirate device. The pirate device therefore fails to decrypt messages encrypted with $g^{rP'(0)}$ and the output of the test is "No". $\square$

From confirmation to tracing: A confirmation test can be used to trace identities of specific traitors. This can be done using the following tracing algorithm:

1. Find a group C for which the confirmation test answers "Yes".
2. Remove an arbitrary member u from C, obtaining C'. ($C = C' \cup \{u\}$.)
3. Run the confirmation test with C' as the subset of suspects.
   - If the test answers "Yes" then reset the decryption device, set $C = C'$ and goto step 2.
   - If the test answers "No" declare that u is a traitor.

Lemma 6 *The tracing algorithm always outputs an identity of a suspected traitor.*

Proof: Denote by C_i the set C that is tested in the tracing algorithm after the ith user is removed. $C_0 = C, \ldots, C_{|C|} = \emptyset$. The algorithm begins with a set C_0 for which the output of the confirmation algorithm is "Yes". For $C_{|C|}$ the confirmation algorithm always answers "No", since $C_{|C|} \cap T = \emptyset$. There is therefore an $1 \leq i \leq |C|$ for which the output of the confirmation algorithm is different for C_{i-1} and C_i. The output of the tracing algorithm is $u \in C_i \backslash C_{i-1}$. □

Lemma 7 *If $C = C' \cup \{u\}$ and the confirmation test answers "Yes" for C and "No" for C', then with high probability u is a traitor (namely $u \in T$).*

Proof: (sketch) Assume to the contrary that $u \notin T$. In this case $C \cap T = C' \cap T$. Therefore the view of the pirate decryption device during the confirmation test is the same whether the input to the test is C or C'. In both cases there is a revocation message containing t values of P' in random locations. The polynomial P' agrees with the values in $C' \cap T$ and conflicts with the other values in T. The output of the confirmation test should therefore be the same in both cases. A contradiction. □

Theorem 8 *Given a group of suspects for which the confirmation algorithm answers "Yes" it is possible to trace the identity of a specific traitor.*

Proof: Given a group of suspects for which the confirmation algorithm answers "Yes" we can run the tracing algorithm. The theorem then follows from lemmas 6 and 7. □

Corollary 9 *There is a black-box tracing algorithm with running time $O(\binom{n}{t})$ that can trace traitors without any a-priori information about their identities.*

Proof: The algorithm tests each subset C of t users using the black-box confirmation algorithm. Since $|T| \leq |C| = t$ there is such an experiment in which $T \subseteq C$ and then the output of the confirmation algorithm is guaranteed to be "Yes". When this event happens the tracing algorithm is run, and it is guaranteed by Theorem 8 to output the identify of an individual traitor. □

3.5 Tracing given a key

Suppose now that the GC is able to "open" a pirate device and reveal the key that the device contains. This enables us to use a deterministic tracing method, based on error-correction codes, which is similar to a tracing method suggested in [4]. This method ensures that if the device key is given in a canonical form (defined below) and was generated using the keys of at most $t/2$ users, then it is possible to identify these users.

Canonical form of keys: The key that a user receives can be defined as a vector $\vec{K_u} = (1, x_u, x_u^2, \ldots, x_u^t)$ and a value y_u that is its inner product with $\vec{a} = (a_0, a_1, \ldots a_t)$, the coefficients of the polynomial P. Any such key allows reconstruction of new group keys that are sent using the revocation scheme. Keys of this exact type are not the only useful keys a small coalition of corrupt users can generate. Consider a coalition of $m \leq t$ users, $\{u_1, u_2, \ldots u_m\}$. Then for any $\vec{b} = (b_0, b_1, \ldots, b_t)$ that is a linear combination of the vectors $\{\vec{K_{u_i}}\}_{i=1}^m$ it is possible for the coalition to compute the inner product of $\vec{b}$ and $\vec{a}$. Such a vector allows reconstructing new keys following revocation messages (assuming not all coalition members were revoked). The coalition cannot generate the inner product of $\vec{a}$ and a vector $\vec{b'}$ that is not a linear combination of the vectors $\{\vec{K_{u_i}}\}_{i=1}^m$, since any result for this inner product is equiprobable given the information known to the coalition.

Definition 10 (Key in canonical form) *A* key in canonical form *is composed of a vector $\vec{b}$ and its inner product with $\vec{a}$, the coefficients of the polynomial.*

It *seems* (although we have no proof for that) that keys in canonical form are the only viable option the pirates can take if they want to generate keys allowing reconstruction. This is stated in the following assumption.

Assumption 11 *A pirate decoder that decrypts messages with non-negligible probability contains a canonical form key.*

Theorem 12 *(Tracing given access to the key of a pirate device) Given a pirate decoder that was generated using at most $t/2$ keys, it is possible to trace the source of at least one key.*

Proof: Based on Assumption 11 the only keys that the decoder can store are in canonical form. Namely, they are a linear combination of at most $t/2$ keys. 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 vectors given to users, find this linear combination. To be more precise assume that the n users are named $u_1, \ldots, u_n$ and consider the following matrix B with n rows and $t+1$ columns:

$$B = \begin{pmatrix} 1 & u_1 & (u_1)^2 & \ldots & (u_1)^t \\ 1 & u_2 & (u_2)^2 & \ldots & (u_2)^t \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & u_n & (u_n)^2 & \ldots & (u_n)^t \end{pmatrix}$$

Define this to be the parity check matrix of a linear code and let the matrix A of size $(n-t-1) \times n$ be the corresponding generating matrix of the code, namely $A \cdot B = 0$. The code is dual to a Generalized Reed-Solomon code and is therefore a Generalized Reed-Solomon code by itself (see Chapter 10.8 in [24]). It can therefore be decoded in polynomial time using the decoding algorithm of Welch and Berlekamp [34].

Now consider a pirate decoding device. Based on Assumption 11 it contains a key in canonical form, generated using the keys of at most $t/2$ users. This key is a vector $\vec{d}$ that is a linear combination of at most $t/2$ rows of the matrix B. Namely, there is a vector $\vec{w}$ of length n, with at most $t/2$ entries different than 0, such that $\vec{w}B = \vec{d}$.

The tracing algorithm is given the vector $\vec{d}$. Its first step is to find an arbitrary vector $\vec{v}$ of length n, such that $\vec{v}B = \vec{d}$. Now, it holds that $(\vec{v} - \vec{w})B = 0$ and therefore $(\vec{v} - \vec{w})$ is in the span of the rows of the matrix A. This means that $(\vec{v} - \vec{w})$ is a codeword of the code generated by the matrix A, and therefore $\vec{v}$ is different from a codeword in at most $t/2$ locations. The tracing algorithm feeds $\vec{w}$ to a decoding algorithm (e.g. the Welsh-Berklekamp algorithm [34]) and finds the locations in which it is different from the codeword (i.e. the error locations). These locations correspond to the non zero entries in the vector $\vec{w}$ and therefore to the identities of the traitors. □

Remark 1 *We do not know how to get full-strength black-box tracing as in [8, 27]. Namely, when all the tracing algorithm gets to examine is the input/output behavior of the pirate-box and the time it has is much smaller than* $\binom{n}{t}$.

Remark 2 *The full version of [4] describes a black-box tracing algorithm against single-key pirates. This algorithm is based on the assumption that the pirate decoder contains only a single convex combination of the keys of the traitors, as well as that it always outputs a decryption of the message sent by the traitor tracing scheme. We do not explore this type of tracing algorithm for our revocation methods.*

3.6 Using the scheme for periodic group key refresh

A very appealing mode of operation of the self enforcement revocation scheme is where 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 that contains sensitive information.

3.7 Public key encryption

A variant of the scheme can be used to enable any party to encrypt messages to the group (even if that party is not a group member), while preserving the revocation, self enforcement,

and tracing properties. It is based on a similar idea to that of the public key tracing scheme of [4].

Initialization: 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 $[0, t]$).

Encryption: Any party can encrypt a message $M \in G_q$ 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 $g^{rP(0)}$. (Note that if it is required to encrypt messages $M \notin G_q$ then the encryption can use $H(g^{rP(0)}) \oplus M$ rather than $g^{rP(0)} \cdot M$, where H is modeled as a random function.)

Revocation: 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:

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

Note that the scheme is identical to revocation method 1 in Section 2.2.2. The value $g^{rP(0)}$ that is used there as the new group key is used here as a key with which the message M is encrypted. In its basic form the public key scheme presented here corresponds to revoking the users with identities $1, \ldots, t$, which do not include any of the real users. If the GC decides to revoke users in the public key scheme presented here then this corresponds to revoking the same users with method 1 of Section 2.2.2. The theorems regarding the security, traceability and self-enforcement of that scheme are therefore valid here too.

3.8 Combining Revocation with Combinatorial Tracing Schemes

Most tracing schemes (such as those in [8, 27]) are based on combinatorial constructions. In these constructions there is a large set of *independently chosen* basic keys. Each user's personal key is a subset of the basic keys. The schemes encrypt messages in a way that 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) and the way the are used by a pirate box reveal at least one of the users in that coalition.

Revocation schemes can be combined with tracing schemes in a multiplicative 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). Tracing is done as in the underlying combinatorial method. Once a user is traced to be a traitor, the basic keys that are included in the user's 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 its keys as well).

Overhead: the bandwidth loss is precisely that of the tracing method. 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 of the tracing method. It is appealing to use our revocation schemes in this scenario, since their storage, communication, and computation overheads 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. We also thank the anonymous referees for useful comments.

References

[1] J. Anzai, N. Matsuzaki and T. Matsumoto, *A Quick Group Key Distribution Scheme with Entity Revocation*. Adv. in Cryptology – Asiacrypt'99, LNCS 1716, Springer, 1999, pp. 333–347.

[2] G.R. Blakley, *Safeguarding cryptographic keys*, AFIPS Conference Proceedings, 48: 313-317, 1979.

[3] D. Boneh, *The Decision Diffie-Hellman Problem*, in Proceedings of the Third Algorithmic Number Theory Symposium, LNCS Vol. 1423, Springer, pp. 48–63, 1998.

[4] D. Boneh and M. Franklin, *An efficient public key traitor tracing scheme*, Adv. in Cryptology – Crypto '99, Springr- LNCS 1666 (1999), 338–353, and a full version available at `http://crypto.stanford.edu/~dabo/pubs.html`.

[5] D. Boneh and J. Shaw, *Collusion-Secure Fingerprinting for Digital date*, Proc. Advances in Cryptology – Crypto '95 (1995), 452–465.

[6] 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.

[7] 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.

[8] B. Chor, A. Fiat and M. Naor, *Tracing Traitors*, Proc. Advances in Cryptology – Crypto '94, Springr-Verlag LNCS 839 (1994), 257–270.

[9] B. Chor, A. Fiat, M. Naor and B. Pinkas, *Tracing Traitors*, IEEE Transactions on Information Theory, Vol. 46, No. 3, pp. 893–910, May 2000.

[10] 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.

[11] H. Cohen, **A course in computational algebraic number theory**, Springer-Verlag, 1996.

[12] 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.

[13] Diffie W. and Hellman M. E., New Directions in Cryptography, *IEEE Trans. on Information Theory*, Nov. 1976, 644-654.

[14] 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.

[15] 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.

[16] P. Feldman, *A practical scheme for non-interactive verifiable secret sharing*, Proc. 28th IEEE Symp. on Foundations of Computer Science, 1987, pp. 427–437.

[17] A. Fiat and M. Naor, *Broadcast Encryption*, Advances in Cryptology – CRYPTO '93, Springer-Verlag LNCS vol. 773, 1994, pp. 480–491.

[18] E. Gafni, J. Staddon and Y. L. Yin, *Efficient methods for integrating traceability and broadcast encryption*, Proc. Advances in Cryptology – Crypto '99, LNCS 1666, Springer, 1999, 372–387.

[19] O. Goldreich, S. Goldwasser and S. Micali, *How to construct random functions*, *J. of the ACM.*, vol. 33, 1986, pp. 792-807.

[20] A. Kiayias and M. Yung, *Self protecting pirates and black-box traitor tracing*, Adv. in Cryptology – Crypto '2001, Springr-Verlag LNCS 2139, pp. 63–79, 2001.

[21] 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.

[22] K. Kurosawa and Y. Desmedt, *Optimum traitor tracing and asymmetric schemes*, Advances in Cryptology – Eurocrypt '98, Springer-Verlag LNCS 1403, pp. 145–157. 1998.

[23] M. Luby, **Pseudo-randomness and applications**, Princeton University Press, 1996.

[24] F. J. MacWilliams and N. J. A. Sloane, **The Theory of Error-Corecting Codes**, North Holland, Amsterdam, 1977.

[25] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, **Handbook of Applied Cryptography**, CRC Press, 1996.

[26] D. Naor, M. Naor and J.B. Lotspiech, *Revocation and Tracing Schemes for Stateless Receivers*, Proc. Advances in Cryptology – Crypto '01, LNCS 2139, Springer-Verlag, 2001, 41–62.

[27] M. Naor and B. Pinkas, *Threshold Traitor Tracing*, Proc. Advances in Cryptology – Crypto '98, LNCS 1462, Springer-Verlag, 502–517, 1998.

[28] 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.

[29] A. Shamir, How to share a secret, *Comm. ACM*, Vol. 22, No. 11, 1979, 612–613.

[30] S. Haber and B. Pinkas, *Combining Public Key Cryptosystems*, Proceedings of the ACM Computer and Security Conference, November 2001.

[31] D.R. Stinson and R. Wei, *Key preassigned traceability schemes for broadcast encryption*, SAC'98, Springer-Verlag LNCS 1556, 1998.

[32] 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.

[33] 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`

[34] L.R. Welch and E.R. Berlekamp, *Error correction for algebraic blockcodes*, U.S. Patent 4633470, issued Dec. 30 1986.

[35] 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.