An Efficient Public Key Traitor Tracing Scheme

(Extended Abstract)

Dan Boneh
dabo@cs.stanford.edu

Matthew Franklin
franklin@parc.xerox.com

Abstract. We construct a public key encryption scheme in which there is one public encryption key, but *many* private decryption keys. If some digital content (e.g., a music clip) is encrypted using the public key and distributed through a broadcast channel, then each legitimate user can decrypt using its own private key. Furthermore, if a coalition of users collude to create a new decryption key then there is an efficient algorithm to trace the new key to its creators. Hence, our system provides a simple and efficient solution to the "traitor tracing problem". Our tracing algorithm is deterministic, and catches *all* active traitors while *never* accusing innocent users, although it is only partially "black box". A minor modification to the scheme enables it to resist an adaptive chosen ciphertext attack. Our techniques apply error correcting codes to the discrete log representation problem.

1 Introduction

Consider the distribution of digital content to subscribers over a broadcast channel. Typically, the distributor gives each authorized subscriber a hardware or software decoder ("box") containing a secret decryption key. The distributor then broadcasts an encrypted version of the digital content. Authorized subscribers are able to decrypt and make use of the content. This scenario comes up in the context of pay-per-view television, and more commonly in web based electronic commerce (e.g. broadcast of online stock quotes or broadcast of proprietary market analysis).

However, nothing prevents a legitimate subscriber from giving a copy of her decryption software to someone else. Worse, she might try to expose the secret key buried in her decryption box and make copies of the key freely available. The "traitor" would thus make all of the distributor's broadcasts freely available to non-subscribers. Chor, Fiat and Naor [4] introduced the concept of a *traitor tracing scheme* to discourage subscribers from giving away their keys. Their approach is to give each subscriber a distinct set of keys that both identify the subscriber and enable her to decrypt. In a sense, each set of keys is a "watermark" that traces back to the owner of a particular decryption box. A coalition of traitors might try to mix keys from many boxes, to create a new pirate box that can still decrypt but cannot be traced back to them. A traitor tracing scheme is "k-collusion resistant" if at least one traitor can always be identified when k of them try to cheat in this way. In practice, especially with tamper-resistant

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 338-353, 1999.

decryption boxes, it may suffice for k to be a fairly small integer, e.g., on the order of 20.

In this paper we present an efficient public key traitor tracing scheme. The public key settings enable anyone to broadcast encrypted information to the group of legitimate receivers. Previous solutions were combinatorial with probabilistic tracing [4, 10, 14–16], and could be either public-key or symmetric-key. Our approach is *algebraic* with *deterministic* tracing, and is inherently public-key. Our approach is much more efficient than the public-key instantiations of previous combinatorial constructions.

Previous approaches [4, 10] incur an overhead that is proportional to the logarithm of the size of the population of honest users. In a commercial setting such as a web broadcast or pay-per-view tv, where the number of subscribers might be in the millions, this is a significant factor. Our approach eliminates this factor. Furthermore, secret keys in our scheme are very short. Each private key is just the discrete log of a single element of a finite field (e.g., as small as 160 bits in practice). The size of an encrypted message is just $2k + 1$ elements of the finite field. The work required to encrypt is about $2k + 1$ exponentiations. Decryption takes far less than $2k + 1$ exponentiations. During decryption, only the final exponentiation uses the private key, which can be helpful when the secret is stored on a weak computational device.

Previous probabilistic tracing methods try to maximize the chance of catching just *one* of the traitors while minimizing the chance of accusing an innocent user. Our tracing method is deterministic. It catches *all* of the traitors who contributed to the attack. Innocent users are never accused, as long as the number of colluders is at or below the collusion threshold. Even when more than k (but less than $2k$) traitors collude, some information about the traitors can be recovered. However, unlike previous tracing methods, our approach is only partially "black box". This limits the tracer in certain scenarios where the pirate decoder can be queried, but the pirate keys cannot be extracted (see Section 4.2).

The intuition behind our system is as follows. Each private key is a different solution vector for the discrete log representation problem with respect to a fixed base of field elements. We can show that the pirate is limited to forming new keys by taking convex combinations of stolen keys. If every set of $2k$ keys is linearly independent, then every convex combination of k keys can be traced uniquely (but not necessarily efficiently). By deriving our keys from a Reed-Solomon code in the appropriate way, we can take advantage of efficient error correction methods to trace uniquely and efficiently. We note that the multi-dimensional discrete log representation problem has been previously used, e.g., for incremental hashing [1] and Signets [6].

Our scheme is traceable if the discrete log problem is hard. The encryption scheme is secure (semantic security against a passive adversary) if the decision Diffie-Hellman problem is hard. A small modification yields security against an adaptive chosen ciphertext attack under the same hardness assumption. That level of protection can be important in distribution scenarios where the decryption boxes (or decryption software) are widely deployed and largely unsupervised.

In Section 2 we give definitions for the traitor tracing problem. Our basic scheme is described in Section 3. The tracing algorithm is detailed in Section 4. Chosen ciphertext security is considered in Section 5. Conclusions are given in Section 7, including an application of our scheme to defending against software piracy.

2 Definitions

For a detailed presentation of the traitor tracing model, see [4]. A public key *traitor tracing* encryption scheme is a public key encryption system in which there is a unique encryption key and multiple decryption keys. The scheme is made up of four components:

Key Generation: The key generation algorithm takes as input a security parameter s and a number ℓ of private keys to generate. It outputs a public encryption key e and a list of private decryption keys $d_1, \ldots, d_\ell$. Any decryption key can be used to decrypt a ciphertext created using the encryption key.

Encryption: The encryption algorithm takes a public encryption key e and a message M and outputs a ciphertext C.

Decryption: The decryption algorithm takes a ciphertext C and any of the decryption keys d_i and outputs the message M. This is an "open" scheme in the sense that only the short decryption keys are secret while the decryption method can be public.

Tracing: Suppose a pirate gets hold of k decryption keys $d_1, \ldots, d_k$. Using the k keys he creates a pirate decryption box (or decryption software) $\mathcal{D}$. The encryption scheme is said to be "k-resilient" if there is a *tracing algorithm* that can determine at least one of the d_i's in the pirate's possession. The tracing algorithm is said to be "black box" if its only use of $\mathcal{D}$ is as an oracle to query on various inputs.

Representations: Our traitor tracing scheme relies on the *representation problem.* When $y = \prod_{i=1}^{2k} h_i^{\delta_i}$ we say that $(\delta_1, \ldots, \delta_{2k})$ is a "representation" of y with respect to the base $h_1, \ldots, h_{2k}$. If $\bar{d}_1, \ldots, \bar{d}_m$ are representations of y with respect to the same base, then so is any "convex combination" of the representations: $\bar{d} = \sum_{i=1}^{m} \alpha_i \bar{d}_i$ where $\alpha_1, \ldots, \alpha_m$ are scalars such that $\sum_{i=1}^{m} \alpha_i = 1$.

3 The encryption scheme

We are now ready to present our tracing traitor encryption scheme. Let s be a security parameter and k be the maximal coalition size. Our scheme defends against any collusion of at most k parties. We wish to generate one public key and ℓ corresponding private keys. Without loss of generality we assume $\ell \geq 2k+2$ (if $\ell < 2k+2$ we set $\ell = 2k+2$ and generate ℓ private keys).

Our scheme makes use of a certain *linear space tracing* code Γ which is a collection of ℓ codewords in $\mathbb{Z}^{2k}$. The construction of the set Γ and the properties it has to satisfy are described in the next section. For now, it suffices to view the ℓ words in Γ as vectors of integers of length $2k$. The set $\Gamma = \{\gamma^{(1)}, \ldots, \gamma^{(\ell)}\}$ is fixed and publicly known.

Let G_q be a group of prime order q. The security of our encryption scheme relies on the difficulty of computing discrete log in G_q. More precisely, the security is based on the difficulty of the Decision Diffie-Hellman problem [3] in G_q as discussed below. One can take as G_q the subgroup of $\mathbb{Z}_p^*$ of order q where p is a prime with $q|p-1$. Alternatively, one can use the group of points of an elliptic curve over a finite field.

Key generation: Perform the following steps:

1. Let $g \in G_q$ be a generator of G_q.
2. For $i = 1, \ldots, 2k$ choose a random $r_i \in \mathbb{Z}_q$ and compute $h_i = g^{r_i}$.
3. The public key is $\langle y, h_1, \ldots, h_{2k} \rangle$, where $y = \prod_{i=1}^{2k} h_i^{\alpha_i}$ for random $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{Z}_q$.
4. A private key is an element $\theta_i \in \mathbb{Z}_q$ such that $\theta_i \cdot \gamma^{(i)}$ is a representation of y with respect to the base $h_1, \ldots, h_{2k}$. The i'th key, θ_i, is derived from the i'th codeword $\gamma^{(i)} = (\gamma_1, \ldots, \gamma_{2k}) \in \Gamma$ by

$$\theta_i = \Big(\sum_{j=1}^{2k} r_j \alpha_j\Big) \Big/ \Big(\sum_{j=1}^{2k} r_j \gamma_j\Big) \pmod{q} \tag{1}$$

   To simplify the exposition we frequently refer to the private key as being the representation $\bar{d}_i = \theta_i \cdot \gamma^{(i)}$. Note however that only θ_i needs to be kept secret since the code Γ is public. One can verify that $\bar{d}_i$ is indeed a representation of y with respect to the base $h_1, \ldots, h_{2k}$.[1]

Encryption: To encrypt a message M in G_q do the following: first pick a random element $a \in \mathbb{Z}_q$. Set the ciphertext C to be

$$C = \langle M \cdot y^a, \ h_1^a, \ \ldots, \ h_{2k}^a \rangle$$

Decryption: To decrypt a ciphertext $C = \langle S, H_1, \ldots, H_{2k} \rangle$ using user i'th secret key, θ_i, compute

$$M = S/U^{\theta_i} \qquad \text{where} \qquad U = \prod_{j=1}^{2k} H_j^{\gamma_j}$$

Here $\gamma^{(i)} = (\gamma_1, \ldots, \gamma_{2k}) \in \Gamma$ is the codeword from which θ_i is derived. The cost of computing U is far less than $2k+1$ exponentiations thanks to simultaneous

[1] A codeword might not have an associated private key in the extremely unlikely event that the denominator is zero in the calculation of θ_i.

multiple exponentiation [9, p. 618]. Also note that U can be computed without knowledge of the private key, leaving only a single exponentiation by the private key holder to complete the decryption.

Before going any further we briefly show that the encryption scheme is sound, i.e. any private key θ_i correctly decrypts any ciphertext. Given a ciphertext $C = \langle M \cdot y^a, h_1^a, \ldots, h_{2k}^a \rangle$, decryption will yield $M \cdot y^a / U^{\theta_i}$ where $U = \prod_{j=1}^{2k} (h_j^a)^{\gamma_j}$. Then

$$U^{\theta_i} = (\prod_{j=1}^{2k} g^{a r_j \gamma_j})^{\theta_i} = (g^{\Sigma_{j=1}^{2k} r_j \gamma_j})^{\theta_i a} = (g^{\Sigma_{j=1}^{2k} r_j \alpha_j})^a = (\prod_{j=1}^{2k} h_j^{\alpha_j})^a = y^a$$

as needed. The third equality follows from Equation (1). More generally, it is possible to decrypt given any representation $(\delta_1, \ldots, \delta_{2k})$ of y with respect to the base $h_1, \ldots, h_{2k}$, since $\prod_{j=1}^{2k} (h_j^a)^{\delta_j} = y^a$.

Tracing algorithm: We describe our tracing algorithm in Section 4.

3.1 Proof of security

We now show that our encryption scheme is semantically secure against a passive adversary assuming the difficulty of the Decision Diffie-Hellman problem (DDH) in G_q. The assumption says that in G_q, no polynomial time statistical test can distinguish with non negligible advantage between the two distributions $D = \langle g_1, g_2, g_1^a, g_2^a \rangle$ and $R = \langle g_1, g_2, g_1^a, g_2^b \rangle$ where g_1, g_2 are chosen at random in G_q and a, b are chosen at random in $\mathbb{Z}_q$.

Theorem 1. *The encryption scheme is semantically secure against a passive adversary assuming the difficulty of DDH in G_q.*

Proof. Suppose the scheme is not semantically secure against a passive adversary. Then there exists an adversary that given the public key $\langle y, h_1, \ldots, h_{2k} \rangle$ produces two messages $M_0, M_1 \in G_q$. Given the encryption C of one of these messages the adversary can tell with non-negligible advantage ϵ which of the two messages he was given. We show that such an adversary can be used to decide DDH in G_q. Given $\langle g_1, g_2, u_1, u_2 \rangle$ we perform the following steps to determine if it is chosen from R or D:

Step 1: Choose random $r_2, \ldots, r_{2k} \in \mathbb{Z}_q$. Set $y = g_1$, $h_1 = g_2$, and $h_i = g_2^{r_i}$ for $i = 2, \ldots, 2k$.
Step 2: Give $\langle y, h_1, \ldots, h_{2k} \rangle$ to the adversary. Adversary returns $M_0, M_1 \in G_q$.
Step 3: Pick a random $b \in \{0, 1\}$ and construct the ciphertext

$$C = \langle M_b u_1, u_2, u_2^{r_2}, \ldots, u_2^{r_{2k}} \rangle$$

Step 4: Give the ciphertext C to the adversary. Adversary returns $b' \in \{0, 1\}$.
Step 5: If $b = b'$ output "D". Otherwise output "R".

Observe that if the tuple $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from D, then the ciphertext C is an encryption of M_b. If the quadruple is from R, then the ciphertext is an encryption of $M_b g_1^{(a_1 - a_2)}$, where $u_1 = g_1^{a_1}$ and $u_2 = g_2^{a_2}$. In other words, the ciphertext is the encryption of a random message. Hence $b = b'$ holds with probability 1/2. By a standard argument, a non-negligible success probability for the adversary implies a non-negligible success probability in deciding DDH. □

3.2 Constructing new representations

To decrypt, it suffices to know any representation of y with respect to the base $h_1, \ldots, h_{2k}$. We have already noted that if $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{Z}_q^{2k}$ are representations of y then any convex combination of $\bar{d}_1, \ldots, \bar{d}_m$ is also a representation of y. The following lemma shows that convex combinations are the only new representations of y that can be efficiently constructed from $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{Z}_q^{2k}$.

Lemma 1. *Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key. Suppose an adversary is given the public key and m private keys $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{Z}_q^{2k}$ for $m < 2k$. If the adversary can generate a new representation $\bar{d}$ of y with respect to the base $h_1, \ldots, h_{2k}$ that is not a convex combination of $\bar{d}_1, \ldots, \bar{d}_m$ then the adversary can compute discrete logs in G_q.*

Proof. Let g be a generator of G_q. Suppose we are given $z = g^x$. We show how to use the adversary to compute x. Choose random $a, b, r_1, \ldots, r_m, s_1, \ldots, s_{2k} \in \mathbb{Z}_q$. Construct the set $\{h_1, \ldots, h_{2k}\}$ where $h_i = z^{r_i} g^{s_i}$ for $1 \leq i \leq m$ and $h_i = g^{s_i}$ for $m+1 \leq i \leq 2k$. Compute $y = z^a g^b$. Find m linearly independent (and otherwise random) solutions $\bar{\alpha}_1, \ldots, \bar{\alpha}_m$ to $\bar{\alpha} \cdot \bar{r} = a \bmod q$. Extend these to be (otherwise random) solutions to $\bar{\alpha} \cdot \bar{s} = b \bmod q$, while keeping the first m entries of each $\bar{\alpha}_i$ unchanged. These m extended vectors are representations of y with respect to the base $h_1, \ldots, h_{2k}$. Suppose that the adversary can find another representation $\bar{\beta}$ that is not a convex combination of $\bar{\alpha}_1, \ldots, \bar{\alpha}_m$. Since it is not a convex combination, we must have that $\beta_1 r_1 + \ldots + \beta_m r_m = a' \neq a \bmod q$. But then $a'x + \bar{\beta} \cdot \bar{s} = ax \bmod q$, which yields the discrete log $x = (\bar{\beta} \cdot \bar{s})(a - a')^{-1} \bmod q$. □

4 Linear space tracing

4.1 The tracing algorithm

We now turn our attention to the tracing algorithm for the encryption scheme of Section 3. Throughout this subsection we assume the pirate decoder contains at least one representation of y. Furthermore, we assume that by examining the decoder implementation it is possible to obtain one of these representations, $\bar{d}$. In Section 4.2 we show that, in some cases, these assumptions are unnecessary.

Suppose the pirate obtains k keys $\bar{d}_1, \ldots, \bar{d}_k$. By Lemma 1, $\bar{d}$ found in the pirate decoder must lie in the linear span of the representations $\bar{d}_1, \ldots, \bar{d}_k$. We construct a tracing algorithm that given $\bar{d}$ outputs one of $\bar{d}_1, \ldots, \bar{d}_k$.

Recall that the construction of private keys made use of a set $\Gamma \subseteq \mathbb{Z}_q^{2k}$ containing ℓ codewords. Each of the ℓ users is given a private key $\bar{d}_i \in \mathbb{Z}_q^{2k}$ which is a multiple of a codeword in Γ. To solve the tracing problem we must construct a set $\Gamma \subseteq \mathbb{Z}_q^{2k}$ containing ℓ codewords with the following property. Let $\bar{d}$ be a point in the linear span of some k codewords $\gamma^{(1)}, \ldots, \gamma^{(k)} \in \Gamma$. Then at least one γ in $\gamma^{(1)}, \ldots, \gamma^{(k)}$ must be a member of any coalition (of at most k users) that can create $\bar{d}$. This γ identifies one of the private keys that *must* have participated in the construction of the pirated key $\bar{d}$. Furthermore, there should exist an efficient tracing algorithm that when given $\bar{d}$ as input, outputs γ. In fact, our tracing algorithm will output every $\gamma^{(i)}$ that has nonzero weight in the linear combination.

The set Γ: We begin by describing the set Γ containing ℓ codewords over $\mathbb{Z}_q^{2k}$. Since q is a large prime we may assume $q > \max(\ell, 2k)$. Consider the following $(\ell - 2k) \times \ell$ matrix:

$$A = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & 2 & 3 & \ldots & \ell \\ 1^2 & 2^2 & 3^2 & \ldots & \ell^2 \\ 1^3 & 2^3 & 3^3 & \ldots & \ell^3 \\ & \vdots & & & \vdots \\ 1^{\ell-2k-1} & 2^{\ell-2k-1} & 3^{\ell-2k-1} & \ldots & \ell^{\ell-2k-1} \end{pmatrix} \pmod{q}$$

Observe that any vector in the span of the rows of A corresponds to a polynomial of degree at most $\ell - 2k - 1$ evaluated at the points $1, \ldots, \ell$.

Let $b_1, \ldots, b_{2k}$ be a basis of the linear space of vectors satisfying $A\bar{x} = 0 \bmod q$. Viewing these $2k$ vectors as the columns of a matrix we obtain an $\ell \times 2k$ matrix B:

$$B = \begin{pmatrix} | & | & | & & | \\ b_1 & b_2 & b_3 & \ldots & b_{2k} \\ | & | & | & & | \end{pmatrix}$$

We define Γ as the set of rows of the matrix B. Hence, Γ contains ℓ codewords each of length $2k$. We note that using Lagrange interpolation one can directly construct the i'th codeword in Γ using approximately ℓ arithmetic operations modulo q.

The tracing algorithm: Consider the set of vectors in Γ. Let $\bar{d} \in \mathbb{Z}_q^{2k}$ be a vector formed by taking a linear combination of at most k vectors in Γ. We show that given $\bar{d}$ one can efficiently determine the unique set of vectors in Γ used to construct $\bar{d}$. Since the vectors in Γ form the rows of the matrix B above we know there exists a vector $\bar{w} \in \mathbb{F}_q^\ell$ of Hamming weight at most k such that $\bar{w} \cdot B = \bar{d}$. We show how to recover the vector $\bar{w}$ given $\bar{d}$.

Step 1: Find a vector $\bar{v} \in \mathbb{F}_q^\ell$ such that $\bar{v} \cdot B = \bar{d}$. Many such vectors exist. Choose one arbitrarily.[2] Since $(\bar{v} - \bar{w}) \cdot B = 0$ we know that $\bar{v} - \bar{w}$ is in

[2] If B is in canonical form with the identity matrix as its first $2k$ rows, then $\bar{v} = (\bar{d} || 0 \ldots 0)$ suffices.

the linear span of the rows of the matrix A (the rows of A span the space of vectors orthogonal to the columns of B). In other words, there exists a unique polynomial $f \in \mathbb{F}_q[x]$ of degree at most $\ell - 2k - 1$ such that $\bar{v} - \bar{w} = \langle f(1), \ldots, f(\ell) \rangle$.

Step 2: Since $\bar{w}$ has Hamming weight at most k, we know that $\langle f(1), \ldots, f(\ell) \rangle$ equals $\bar{v}$ in all but k components. Hence, using Berlekamp's algorithm [2] we can find f from $\bar{v}$. The polynomial f gives us the vector $\bar{v} - \bar{w}$ from which we recover $\bar{w}$ as required.

For completeness we briefly recall Berlekamp's algorithm. The algorithm enables us to find f given the vector $\bar{v} \in \mathbb{Z}_q^\ell$. Let g be a polynomial of degree at most k such that $g(i) = 0$ for all $i = 1, \ldots, \ell$ for which $f(i) \neq v_i$ (where v_i is the i'th component of $\bar{v}$). Then we know that for all $i = 1, \ldots, \ell$ we have $f(i)g(i) = g(i)v_i$. The polynomial fg has degree at most $\ell - k - 1$. Hence, we get ℓ equations (for each of $i = 1, \ldots, \ell$) in ℓ variables (the variables are the coefficients of the polynomials fg and g, where the leading coefficient of g is 1). Let h and g be a solution where g is a non-zero polynomial: h is a polynomial of degree at most $\ell - k - 1$ and g is of degree at most k. We know that whenever $f(i) = v_i$ (i.e. at $\ell - k$ points) we have $h(i) = g(i)v_i = g(i)f(i)$. It follows that $f = h/g$.

This completes the description of the tracing algorithm. Our tracing algorithm satisfies several properties:

Full tracing Given a pirated key $\bar{d}$ the tracing algorithm will recover *all* keys that were used in the construction of $\bar{d}$. In all previous tracing schemes only one pirate was guaranteed to be found. Note that the set of traced pirates may be a subset of the guilty coalition, i.e., the coalition may have used only a subset of the keys at its disposal to create $\bar{d}$. All keys that were actually used (i.e. the pirated key could not be constructed without them) will be found.

Error free tracing The tracing algorithm is deterministic in the sense that there is no error probability. Any key output by the tracing algorithm must have participated in the construction of the pirated key.

Beyond threshold tracing If more than k parties colluded to create the pirated key, then Berlekamp's algorithm may fail to recover the polynomial f, and tracing will fail. Above this bound, recent results of Guruswami and Sudan [7] may be used to output a list of candidate polynomials for f. The tracer gets a list of "leads" for the fraud investigation that includes the actual colluders. This will be effective against coalitions of size at most $2k - 1$.

Running time The tracing algorithm requires that we solve a linear system of dimension ℓ (the total number of users). A naive implementation runs in time $O(\ell^2)$ (field operations). Asymptotically efficient versions of Berlekamp's algorithm run in time $\tilde{O}(\ell)$, where the "soft-Oh" notation hides polylog terms. The fastest known algorithm, due to Pan [12], runs in time $O(\ell \log \ell \log\log \ell)$.

4.2 Black box tracing

The question that remains is the following: given a pirate decryption box (or pirated decryption software) how does one recover the representation $\bar{d}$ used by the box? It is conceivable that a pirate box could decrypt while the tracer is unable to extract any representation from it. If the pirate box is tamper-resistant, the tracer might be unable to open it. Even after seeing the decoding logic, the tracer might be unable to efficiently deduce from it a valid representation.

The above discussion shows that it is desirable to enable the tracer to extract a key used by the box simply by observing its behavior on a few chosen ciphertexts. In other words, the tracing algorithm may only use the pirated box as an oracle. Given this oracle the tracing algorithm must output one of the keys at the pirate's possession. Previous traitor tracing schemes support this kind of black box tracing [4, 10]. Our scheme supports two types of black box tracing techniques. The first is very efficient, but assumes the pirate is restricted in how it constructs the decryption box. The second works against an arbitrary pirate, but is less efficient.

Single-key pirates. A natural strategy for a pirate to build a pirate decryption box is to form a new representation $\bar{d}$ and then create a box that decrypts using this representation. We call this a "single-key pirate" since only a single representation of y is embedded in the pirate decoder. We will show an efficient black box tracing algorithm that works against a single-key pirate. Note that when a single user attempts to construct a decoder that cannot be traced back to him he is essentially acting as a single-key pirate.

We model the single-key pirate's behavior as if he is divided into two distinct parties. The first party is given the k keys $\bar{d}_1, \ldots, \bar{d}_k$, and creates some new key $\bar{d}$. By Lemma 1 we know $\bar{d}$ must be a convex combination of the k given keys. The first party then hands $\bar{d}$ to the second party. The second party, seeing only $\bar{d}$ and the public key, is free to implement the decryption box however he wants. For convenience, we refer to the second party as the "box-builder".

We show how the tracer can extract the representation $\bar{d} = (\delta_1, \ldots, \delta_{2k})$ from a decoder created by a single-key pirate. The basic idea is to observe the decoder's behavior on *invalid* ciphertexts, e.g., $\tilde{C} = \langle S, h_1^{z_1}, \ldots, h_{2k}^{z_{2k}} \rangle$, where the non-constant vector $\bar{z}$ is chosen by the tracer. This ciphertext is invalid since the h_i's are raised to different powers. The next lemma shows that the pirate box cannot distinguish invalid ciphertexts from valid ciphertexts (assuming the difficulty of DDH in G_q). Hence, on input $\tilde{C} = \langle S, H_1, \ldots, H_{2k} \rangle$ it must respond with A where

$$A = S / \prod H_i^{\delta_i} = S / \prod h_i^{z_i \delta_i}$$

Lemma 2. *Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key and let $\bar{d} = (\delta_1, \ldots, \delta_{2k})$ be a representation of y. Suppose that given $\bar{d}$ and the public key the box-builder is able to construct a pirate decoder that will correctly decrypt all valid ciphertexts, but when given a random invalid ciphertext $\tilde{C} = \langle S, h_1^{z_1}, \ldots, h_{2k}^{z_{2k}} \rangle$ will output a value different from $S / \prod h_i^{z_i \delta_i}$, with non-negligible probability (over the choice of S and $\bar{z}$). Then the box-builder can be used to solve DDH in G_q.*

Proof Sketch. Given a challenge tuple $\langle g_1, g_2, u, v\rangle$ we decide whether it is a random tuple or a Diffie-Hellman tuple as follows: build a public key $\langle y, h_1, \ldots, h_{2k}\rangle$ by picking random $a_i, b_i \in \mathbb{Z}_q$ for $i = 1, \ldots, 2k$ and setting $h_i = g_1^{a_i} g_2^{b_i}$. The element $y \in G_q$ is constructed as in the key generation algorithm, i.e. $y = \prod h_i^{\alpha_i}$ for random $\alpha_i \in \mathbb{Z}_q$. Next, build a ciphertext

$$\tilde{C} = \langle S,\ u^{a_1} v^{b_1},\ \ldots,\ u^{a_{2k}} v^{b_{2k}}\rangle$$

where S is random in G_q. Observe that if the challenge $\langle g_1, g_2, u, v\rangle$ is a Diffie-Hellman tuple then $\tilde{C}$ is a random valid ciphertext. Otherwise, $\tilde{C}$ is a random invalid ciphertext. Next, we ask the box-builder to build a pirate decoder given $(\alpha_1, \ldots, \alpha_{2k})$ and the public key. We then feed $\tilde{C}$ into the pirate decoder. Since the decoder behaves differently for valid and invalid ciphertexts the result enables us to solve the given DDH challenge. □

By querying at invalid ciphertexts the tracer learns the value $\prod h_i^{z_i \delta_i} = S/A$ for vectors $\bar{z}$ of its choice. After $2k$ queries with random linearly independent $\bar{z}$, the tracer can solve for $h_1^{\delta_1}, \ldots, h_{2k}^{\delta_{2k}}$. Since the tracer knows the discrete log of the h_i's base g (recall Step 2 of key generation) it can compute $g^{\delta_1}, \ldots, g^{\delta_{2k}}$. Ideally, we would like to use homomorphic properties of the discrete log to run the tracing algorithm of the previous section "in the exponents". Unfortunately, it is an open problem to run Berlekamp's algorithm this way. Instead, we can recover the vector $\bar{d} = (\delta_1, \ldots, \delta_{2k})$ from $\langle g^{\delta_1}, \ldots, g^{\delta_{2k}}\rangle$ by using recent results on trapdoors of the discrete log [11, 13] modulo $p^2 q$ and modulo N^2. For instance, the trapdoor designed by Paillier [13] shows that if encryption is done in the group $\mathbb{Z}_{N^2}^*$ then the secret factorization of N (known to the tracer only) enables the tracer to recover $\langle \delta_1, \ldots, \delta_{2k}\rangle$ mod N from $\langle g^{\delta_1}, \ldots, g^{\delta_{2k}}\rangle$. The tracing algorithm of the previous section can now be used to recover the keys at the pirate's possession. This completes the description of the black box tracing algorithm for single-key pirates.

Arbitrary pirates. Unfortunately, pirates do not have to limit themselves to a single-key strategy. For instance, the pirate could embed multiple representations of y in the pirate decoder. The decoder could use a different random convex combination of its representations each time it decrypts. This would defeat the approach described above. It is an open problem to build an extractor that will efficiently extract some representation in the convex hull of the keys given to the pirate.

To achieve black box tracing against an arbitrary pirate we rely on "black box confirmation". Suppose the tracer is given an arbitrary pirate decoder, and the tracer suspects a particular set T of at most k traitors. By querying the pirate decoder in a black box fashion, the tracer will be able to efficiently verify this suspicion with high probability. More precisely, the tracer will be able to determine that the pirate must possess some unknown subset of the keys of members of T.

Let $\bar{d}_1, \ldots, \bar{d}_k$ be the keys belonging to members of T and let g be a generator of G_q. To confirm its suspicion of T the tracer queries the decoder with an invalid ciphertext $\tilde{C} = \langle S, g^{z_1}, \ldots, g^{z_{2k}} \rangle$, where the vector $\bar{z}$ satisfies $\bar{z} \cdot \bar{d}_i = w$ for all $i \in T$. Here w is a random element of G_q. As in Lemma 2, the decoder cannot distinguish this invalid ciphertext from a real one. Consequently, it will respond with $A = S / \prod g^{z_i \delta_i}$ where $\langle \delta_1, \ldots, \delta_{2k} \rangle$ is some representation of y. If T is indeed the coalition that created the decoder, then by Lemma 1 we know that $\langle \delta_1, \ldots, \delta_{2k} \rangle$ is in the convex hull of $\bar{d}_1, \ldots, \bar{d}_k$. Hence, if T is the guilty coalition we know that $A = S/g^w$. Confidence in this test can be increased by making multiple queries, where each query is constructed independently using different $S, \bar{z}, w$. If for a suspect coalition T the pirate decoder always responds with $A = S/g^w$ then the pirate must possess a subset of the keys belonging to T. Note that this confirmation algorithm does not require trapdoors of the discrete log.

Since we are able to do black box confirmation, we can do black box tracing by running the confirmation algorithm on all $\binom{n}{k}$ candidate coalitions (n is the total number of users in the system). This results in an inefficient tracing algorithm, but it shows that black box tracing is possible in principle.

5 Chosen ciphertext security

In a typical scenario where our system is used it is desirable to defend against chosen ciphertext attacks. Fortunately, our scheme can be easily modified to be secure against adaptive attacks. The modification is similar to the approach used by Cramer and Shoup [5]. As in Section 3 we work in a group G_q of prime order q. For example, G_q could be a subgroup of order q of $\mathbb{Z}_p^*$ for some prime p where $q | p-1$.

Key generation: Let g be a generator of G_q. Pick random $r_1, \ldots, r_{2k} \in \mathbb{Z}_q$ and set $h_i = g^{r_i}$ for $i = 1, \ldots, 2k$. Next, we pick random $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{Z}_q$ and compute

$$y = h_1^{\alpha_1} h_2^{\alpha_2} \cdots h_{2k}^{\alpha_{2k}} \; ; \quad c = h_1^{x_1} h_2^{x_2} \; ; \quad d = h_1^{y_1} h_2^{y_2}$$

The public key is $\langle y, c, d, h_1, \ldots, h_{2k} \rangle$. The private key is as in Section 3, but also includes $\langle x_1, x_2, y_1, y_2 \rangle$. Hence, user i's private key is $\langle \theta_i, x_1, x_2, y_1, y_2 \rangle$.

Encryption: To encrypt a message $M \in G_q$ do the following: pick a random element $a \in \mathbb{Z}_q$, and compute

$$S = M \cdot y^a \; ; \quad H_1 = h_1^a \; , \; \ldots \; , \; H_{2k} = h_{2k}^a$$

$$\nu = \mathcal{H}(S, H_1, \ldots, H_{2k}) \; ; \quad v = c^a d^{a\nu}$$

where $\mathcal{H}$ is a collision resistant hash function (or chosen from a family of universal one-way hash functions). Set the ciphertext C to be

$$C = \langle S, H_1, \ldots, H_{2k}, v \rangle$$

It is a bit surprising that the system can be made secure against chosen ciphertext attacks by appending a single element v to the ciphertext.

Decryption: To decrypt a ciphertext $C = \langle S, H_1, \ldots, H_{2k}, v\rangle$ using a private key $\langle \theta_i, x_1, x_2, y_1, y_2\rangle$ first compute $\nu = \mathcal{H}(S, H_1, \ldots, H_{2k})$ and check that

$$H_1^{x_1+y_1\nu} \cdot H_2^{x_2+y_2\nu} = v$$

If the test fails, reject the ciphertext. Otherwise, output

$$M = S/U^{\theta_i} \qquad \text{where} \qquad U = \prod_{j=1}^{2k} H_j^{\gamma_j}$$

and $\gamma^{(i)} = (\gamma_1, \ldots, \gamma_{2k}) \in \Gamma$ is the codeword from which θ_i is derived.

Tracing: The tracing algorithm remains unchanged.

We show that the scheme is secure against adaptive chosen ciphertext attack. In other words, we show that the scheme is secure in the following environment: an adversary is given the public key. It generates two messages M_0, M_1 and is given the encryption $C = E(M_b)$ for $b \in \{0, 1\}$ chosen at random. The adversary's goal is to predict b. To do so he is allowed to interact with a decryption oracle that will decrypt any valid ciphertext other than C. If the adversary's guess for b is b' and the probability that $b = b'$ is $\frac{1}{2} + \epsilon$ then we say that the adversary has advantage ϵ. The system is said to be secure against an adaptive chosen ciphertext attack if the adversary's advantage in predicting b is negligible (as a function of the security parameter).

Theorem 2. *The above cryptosystem is secure against an adaptive chosen ciphertext attack assuming that (1) the Decision Diffie-Hellman problem is hard in the group G_q, and (2) the hash function $\mathcal{H}$ is collision resistant (or chosen from a family of universal one-way hash functions).*

We assume the hash function $\mathcal{H}$ is collision resistant. Suppose there exists a polynomial time adversary $\mathcal{A}$ that is able to obtain a non-negligible advantage in predicting b when the above cryptosystem is used. We show that $\mathcal{A}$ can be used to solve the Decision Diffie-Hellman problem in G_q.

Given a tuple $\langle g_1, g_2, u_1, u_2\rangle$ in G_q we perform the following steps to determine if it is a random tuple (i.e chosen from R) or a Diffie-Hellman tuple (i.e chosen from D):

Init Set $h_1 = g_1$ and $h_2 = g_2$. pick random $r_3, \ldots, r_{2k} \in \mathbb{Z}_q$ and set $h_i = g_2^{r_i}$ for $i = 3, \ldots, 2k$. Next, choose random $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{Z}_q$ and compute

$$y = h_1^{\alpha_1} \cdots h_{2k}^{\alpha_{2k}} \; ; \quad c = h_1^{x_1} h_2^{x_2} \; ; \quad d = h_1^{y_1} h_2^{y_2}$$

Challenge The adversary $\mathcal{A}$ is given the public key and outputs two messages $M_0, M_1 \in G_q$. We pick a random $b \in \{0,1\}$ and compute:

$$S = M_b \cdot u_1^{\alpha_1} u_2^{\alpha_2} \prod_{i=3}^{2k} u_2^{\alpha_i r_i}$$

$$H_1 = u_1\ , \quad H_2 = u_2\ , \quad H_3 = u_2^{r_3}\ , \quad \ldots\ , \quad H_{2k} = u_2^{r_{2k}}$$

$$\nu = \mathcal{H}(S, H_1, \ldots, H_{2k})\ ; \qquad v = u_1^{x_1 + y_1 \nu} u_2^{x_2 + y_2 \nu}$$

The challenge ciphertext given to $\mathcal{A}$ is $C = \langle S, H_1, \ldots, H_{2k}, v \rangle$.

Interaction When the adversary $\mathcal{A}$ asks to decrypt a ciphertext

$$C' = \langle S', H'_1, \ldots, H'_{2k}, v' \rangle$$

we respond as in a normal decryption: first we check validity of the ciphertext and reject invalid ciphertexts. For valid ciphertext we give $\mathcal{A}$ the plaintext $M = S' / \prod_{j=1}^{2k} (H'_i)^{\alpha_j}$.

Output Eventually the adversary $\mathcal{A}$ outputs a $b' \in \{0,1\}$. If $b = b'$ we say the input tuple is from D otherwise we say R.

This completes the description of the algorithm for deciding DDH using $\mathcal{A}$. To complete the proof of Theorem 2 it remains to show two things:

- When $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from D the joint distribution of the adversary's view and the bit b is statistically indistinguishable from the actual attack.
- When $\langle g_1, g_2, u_1, u_2 \rangle$ is chosen from R the hidden bit b is (essentially) independent of the adversary's view.

The proofs of both statements are similar to the proofs given by Cramer and Shoup [5] and will be given in the full version of the paper. Based on the two statements a standard argument shows that if the adversary $\mathcal{A}$ has advantage ϵ in predicting b then the above algorithm for deciding DDH also has advantage ϵ. This completes the proof of Theorem 2.

Key extraction and black box tracing. It is surprising that although the scheme is resistant to chosen ciphertext attack, the decryption box will decrypt invalid ciphertexts. In particular, it will decrypt an invalid ciphertext $\tilde{C} = \langle S, H_1, \ldots, H_{2k}, v \rangle$ where

$$S = M \cdot y^a\ ; \quad H_1 = h_1^a\ , \quad H_2 = h_2^a\ , \quad H_3 = h_3^{b_3}\ , \quad H_4 = h_4^{b_4}\ , \ldots\ , \quad H_{2k} = h_{2k}^{b_{2k}}$$

$$\nu = \mathcal{H}(S, H_1, \ldots, H_{2k})\ ; \quad v = c^a d^{a\nu}$$

This is an invalid ciphertext since the h_i's are raised to different powers. It passes the decryptor's test since h_1 and h_2 are raised to the same power. It cannot be distinguished from a valid ciphertext, assuming the hardness of DDH in G_q. The ideas of Section 4.2 can then be applied for black box tracing in this setting.

6 Pitfalls in designing public key traitor tracing schemes

The pirate decoder need not work in the same way as a legitimate decoder in a traitor tracing scheme. For example, the pirate may be able to derive a short string w that enables decryption but is not a legitimate decryption key. If w could be derived from *any* subset of k private keys, then it would be impossible to trace. This successful pirate strategy can be shown to defeat one of the traitor tracing schemes proposed by Kurosawa and Desmedt in [8]. Their scheme works as follows:

Key generation: Let g be a generator of a group G_q of order q for some prime q. Choose a random polynomial $f(x) = a_0 + a_1 x + \ldots + a_k x^k$ over $\mathbb{Z}_q$. Compute

$$y_0 = g^{a_0}, \quad \ldots, \quad y_k = g^{a_k}$$

The public key is $\langle g, y_0, \ldots, y_k \rangle$. The private key for user i is $d_i = f(i)$.

Encryption: To encrypt a message $M \in G_q$ compute

$$C = \langle g^r,\ M \cdot y_0^r,\ y_1^r,\ \ldots,\ y_k^r \rangle$$

where r is random in $\mathbb{Z}_q$.

Decryption: To decrypt a ciphertext $C = \langle a, b_0, \ldots, b_k \rangle$ using user i's private key d_i compute:

$$\left(b_0 \cdot b_1^i \cdot b_2^{(i^2)} \cdots b_k^{(i^k)} \right) / a^{f(i)} = M$$

The authors show that given the public key and k private keys $f(i_1), \ldots, f(i_k)$ it is impossible to construct another private key $f(j)$ for user j, unless the discrete log problem in G_q is easy. However, let $w = (u, w_0, w_1, \ldots, w_k)$ be any convex combination of the vectors $v_1, \ldots, v_k$, defined by:

$$v_1 = \left[f(i_1),\ 1,\ i_1,\ i_1^2,\ \ldots, i_1^k \right]$$

$$\vdots \qquad\qquad \vdots$$

$$v_k = \left[f(i_k),\ 1,\ i_k,\ i_k^2,\ \ldots, i_k^k \right]$$

Then w is not a legitimate private key, but it can be used to decrypt any ciphertext $C = \langle a, b_0, \ldots, b_k \rangle$ since

$$b_0^{w_0} \cdots b_k^{w_k} / a^u = M$$

One can show that many of the convex combinations w cannot be traced to any traitor. To do so, one shows the existence of disjoint coalitions of size k that can all create the same w. This example illustrates the importance of black box tracing, to ensure that it is possible to trace no matter how the decoder is implemented. By increasing the degree of f from k to $2k$, it should be possible to demonstrate k-resilient black box confirmation (and thus inefficient black box tracing) for this scheme as in Section 4.2.

7 Conclusion

We present an efficient public key solution to the traitor tracing problem. Our construction is based on Reed-Solomon codes and the representation problem for discrete logs. Traceability follows from the hardness of discrete log. The semantic security of the encryption scheme against a passive attack follows from the Decision Diffie-Hellman assumption. A simple extension achieves security against an adaptive chosen ciphertext attack under the same hardness assumption. The private key in all cases is just a single element of a finite field and can be as short as 160 bits. The cryptosystem can be made to work in any group in which the Decision Diffie-Hellman problem is hard. It is an interesting open question to improve on the "black box" traceability of our approach. Also, it seems reasonable to believe that there exists an efficient public key tracing traitors scheme that is completely collusion resistant. In such a scheme, any number of private keys cannot be combined to form a new key. Similarly, the complexity of encryption and decryption is independent of the size of the coalition under the pirate's control. An efficient construction for such a scheme will provide a useful solution to the public key tracing traitors problem.

To conclude, we mention an application of our system to defending against software piracy. Typically, when new software is installed from a CD-ROM the user is asked to enter a short unique key printed on the CD cover. This key identifies the installed copy. Clearly the key printed on the CD cover has to be short (say under 20 characters) since it is typed in manually. Our system can be used in this settings as follows: since our private key can be made 120 bits long (to achieve 2^{60} security) it can be printed on the CD cover (each character encodes 6 bits). The software on the CD is encrypted using our system's public key. When the user types in his unique CD key the software is decrypted and installed on the user's machine. However, if a software pirate attempts to create illegal copies of the distribution CD (say using a CD-ROM burner) he must also attach a short printed key to the disk. Using our system, the key he attaches to the bootlegged copies can be traced back to him.

Acknowledgments

The second author would like to thank his colleagues at Xerox PARC for helpful discussions.

References

1. M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing", in proc. Crypto '94, 216–233.
2. L. Welch and E. Berlekamp, "Error correction of algebraic block codes", U.S. Patent No. 4,633,470, issued December 1986.
3. D. Boneh, "The decision Diffie-Hellman problem", In proc. of the Third Algorithmic Number Theory Symposium (ANTS), LNCS, Vol. 1423, Springer-Verlag, pp. 48–63, 1998.

4. B. Chor, A. Fiat and M. Naor, "Tracing traitors", in proc. Crypto '94, pp. 257–270.
5. R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", in proc. Crypto '98, pp. 13–25.
6. C. Dwork, J. Lotspiech and M. Naor, "Digital signets: self-enforcing protection of digital information", in proc. of STOC '96, pp. 489–498, 1996.
7. V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes" in proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS), 1998.
8. K. Kurosawa, and Y. Desmedt, "Optimum traitor tracing and asymmetric schemes", in proc. of Eurocrypt '98, pp. 145–157.
9. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
10. M. Naor and B. Pinkas, "Threshold traitor tracing", in proc. Crypto '98, pp. 502–517.
11. T. Okamoto, S. Uchiyama, "A new public key cryptosystem as secure as factoring", in proc. Eurocrypt '98, pp. 308–318.
12. V. Pan, "Faster solution of the key equation for decoding BCH error-correcting codes" in proc. 29th ACM Symposium on Theory of Computation, pp. 168–175, 1997.
13. P. Paillier, "Public-Key Cryptosystems Based on Discrete Logarithm Residues", in proc. Eurocrypt '99, pp. 223–238.
14. B. Pfitzmann, "Trials of traced traitors", in proc. Information Hiding Workshop, 49–64, 1996.
15. B. Pfitzmann and M. Waidner, "Asymmetric fingerprinting for larger collusions", in proc. ACM Conference on Computer and Communication Security, 151–160, 1997.
16. D. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes", *SIAM Journal on Discrete Math*, 11(1), 41–53, 1998.