An Efficient Public Key Traitor Tracing Scheme

Dan Boneh
dabo@cs.stanford.edu

Matthew Franklin
franklin@cs.ucdavis.edu

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". 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 [5] 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 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 [5, 11, 15, 16, 17], 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. One earlier construction was algebraic in nature [9], although it was later determined to be insecure [18].

Previous approaches [5, 11] 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. Innocent users are never accused, as long as the number of colluders is at or below the collusion threshold, and as long as the complexity assumption remains true. Even when more than k (but less than $2k$) traitors collude, some information about the traitors can be recovered.

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 [7].

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 and a non-black-box tracing algorithm is described in Section 3. Black box tracing against an arbitrary pirate is detailed in Section 4. Black box tracing against a restrictive "single-key pirate" is given in Section 5. Chosen ciphertext security is considered in Section 6, and an open problem is discussed in Section 7. Conclusions are given in Section 8, including an application of our scheme to defending against software piracy.

2 Definitions

For a detailed presentation of the traitor tracing model, see [5]. 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, a number ℓ of private keys to generate, and a number k which we call the collusion bound. It outputs a public encryption key e and a list of distinct 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.

Tracing: The tracing algorithm is given as input all random bits used during key generation (which can be viewed as a "tracing key") as well as a pirate decoder $\mathcal{D}$, say, given as a boolean circuit. It outputs an index $i \in \{1, \ldots, \ell\}$ or `fail`. 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.

We model the pirate as an algorithm that is given the public key and k random decryption keys $d_1, \ldots, d_k$ from the set of all ℓ keys. Using the public key and the k private keys, the pirate creates a pirate decryption box (or decryption software) $\mathcal{D}$. The pirate box $\mathcal{D}$ must correctly decrypt all but a negligible fraction of the valid ciphertexts generated by the encryption algorithm.

At a high level, the encryption scheme is said to be "k-resilient" if the *tracing algorithm*, given $\mathcal{D}$ created by some pirate, and all random bits used during key generation, determines at least one of the d_i's in the pirate's possession used to create $\mathcal{D}$. See Appendix A for a precise definition.

Clearly black box tracing is preferable to tracing algorithms that rely on information embedded in the implementation of $\mathcal{D}$. After all, extracting information from the implementation of $\mathcal{D}$ might be difficult: the software executable might be obfuscated, or $\mathcal{D}$ might be implemented in a temper resistant device. Black box tracing is also important for proving that we do not assume the pirate must embed specific information in the pirate decoder.

Black box tracing. We describe black box tracing in more detail. One notion of black box tracing is that the tracer uses the pirate decoder as a decryption oracle. Given a string C the pirate decoder outputs either (1) "`invalid`" meaning C is not a valid ciphertext, or (2) outputs a plaintext M which it claims is the decryption of C. In this model the tracer sees the full output of the pirate decoder, namely the decryption of C. In some settings such full access to the pirate decoder may not be possible. For example, consider a tamper resistant pirate decoder intended for playing music. The decoder is given an encrypted music clip and either plays the music, or says that the given music clip is invalid. When the tracer queries the pirate decoder its only feedback is whether the decoder plays the music or outputs an error. In other words, all the tracer learns is whether the decoder was successful in decrypting the given ciphertext or not. The tracer does not see the full output of the pirate decryption algorithm. Hence, we can define two types of black box tracing:

Full access black box tracing: In this model the tracer issues a query C to the pirate decoder. The decoder returns either (1) a plaintext M which is supposedly the decryption of C, or (2) it returns `invalid` meaning the ciphertext C is invalid. The decoder can behave maliciously and return either one of the above for any query. When the decoder returns a message M it is free to choose the message maliciously. However, if C is a well formed ciphertext the decoder must return the decryption of C.

Minimal access black box tracing: In this model a query to the pirate decoder is a pair $\langle C, M \rangle$ where C is some ciphertext query and M is some plaintext. The query is given to some third party who then runs the decoder on input C. If the decoder outputs M the third party returns `valid` meaning that C is a valid encryption of M. Otherwise, it returns `invalid` meaning C is not. Thus, the third party severely limits the tracer's access to the pirate decoder. We argue below that this is the correct model for black-box tracing. As in the full access case, the decoder can behave maliciously, but if C is a well formed encryption of M then the decoder must return M so that third party returns `valid`. A precise security definition as a game between a challenger and an adversary is given in Appendix A.

In Section 4 and Section 5 we give two black box tracing algorithms. The first only requires minimal black box access to the decoder. The second requires full black box access.

Our definition of minimal access fits well with the music player analogy. Consider a music player that takes an encrypted music clip and either plays the music or says that the clip is invalid. The music player must play all well formed music clips. A well formed music clip is formated as $\langle C, E_K(S)\rangle$ where E is a semantically secure encryption scheme, K is a symmetric key used to encrypt the music S, and C is a proper encryption of K using the traitor tracing system. Suppose the traitor tracing scheme supports minimal access black box tracing. We trace a pirate music player as follows: when the tracer issues a query $\langle C, K\rangle$ we pick a random piece of music S and create the music clip $\langle C, E_K(S)\rangle$. If the player plays the music clip we respond to the tracer query with `valid`. Otherwise, we respond with `invalid`. The tracing algorithm will correctly identify one of the pirate keys.

Throughout the paper we assume the pirate boxes are stateless. That is, the pirate decoder does not respond to decryption requests based on its responses to previous requests. This does not seem to be a problem since the scheduling of tracing queries can be randomized, and furthermore, tracer queries can be interspersed within regular decryption requests.

We point out that our definition of the pirate says that the pirate obtains a *random* subset of size k of the ℓ private keys. The pirate does not choose which private keys he receives. This is a a natural definition since we are assuming that keys are assigned to users in a random fashion: when a new user buys a decryption box the user receives a box containing a randomly chosen unassigned private key. Hence, even if the pirate breaks into boxes belonging to k users *of his choice*, the pirate still obtains a set of k random keys.

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 $\sum_{i=1}^{m} \alpha_i = 1$.

Notation: We let $\bar{u} \cdot \bar{v}$ denote the inner product of the two vectors $\bar{u}$ and $\bar{v}$.

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

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

   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 $h_1, \dots, h_{2k}$.[1]

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

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

Decryption: To decrypt a ciphertext $C = \langle S, H_1, \dots, H_{2k} \rangle$ 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 $\gamma^{(i)} = (\gamma_1, \dots, \gamma_{2k}) \in \Gamma$ is the codeword from which θ_i is derived. The cost of computing U is far less than $2k+1$ exponentiations thanks to simultaneous multiple exponentiation [10, 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 $C = \langle M \cdot y^a, h_1^a, \dots, h_{2k}^a \rangle$, 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 $(\delta_1, \dots, \delta_{2k})$ of y with respect to the base $h_1, \dots, h_{2k}$, since $\prod_{j=1}^{2k} (h_j^a)^{\delta_j} = y^a$.

[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.

Tracing algorithm: We describe our tracing algorithm in Section 3.3.

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 $\mathbb{F}_q$.

Theorem 3.1. *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 $r_2, \ldots, r_{2k} \in \mathbb{F}_q$. 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. $\square$

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 $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{F}_q^{2k}$ 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 $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{F}_q^{2k}$.

Lemma 3.2. *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 $\bar{d}_1, \ldots, \bar{d}_m \in \mathbb{F}_q^{2k}$ for $m < 2k - 1$. 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 $b, r_1, \ldots, r_{2k}, s_1, \ldots, s_{2k} \in Z_q$. Construct the set $\{h_1, \ldots, h_{2k}\}$ where $h_i = z^{r_i} g^{s_i}$ for all i, $1 \le i \le 2k$. Compute $y = g^b$. Find m linearly independent (and otherwise random) solutions $\bar{\alpha}_1, \ldots, \bar{\alpha}_m$ to $\bar{\alpha} \cdot \bar{r} = 0 \bmod q$ and $\bar{\alpha} \cdot \bar{s} = b \bmod q$. These m 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 $\bar{\alpha}_1, \ldots, \bar{\alpha}_m$. Then $\bar{\beta} \cdot \bar{r} \neq 0 \bmod q$ with probability at least $1 - \frac{1}{q}$. But then $(b - \bar{\beta} \cdot \bar{s})(\bar{\beta} \cdot \bar{r})^{-1}$ is the discrete log of z. □

3.3 Non-Black-Box Tracing

We now turn our attention to a tracing algorithm for our basic encryption scheme. 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, $\bar{d}$. Hence, the tracing algorithm presented in this section is not black box: it assumes one can extract at least one representation of y from the pirate decoder. Whenever it is easy to reverse engineer the pirate decoder this algorithm directly exposes the keys used by the pirate. In the next section we show how to convert this algorithm into a black-box tracing algorithm that exposes the pirate keys without reverse engineering the pirate decoder.

Suppose the pirate obtains k keys $\bar{d}_1, \ldots, \bar{d}_k$. Let $\bar{d}$ be the representation of y found in the pirate decoder. Then by Lemma 3.2 we know that $\bar{d}$ 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 $\Gamma \subseteq \mathbb{F}_q^{2k}$ containing ℓ codewords. Each of the ℓ users is given a private key $\bar{d}_i \in \mathbb{F}_q^{2k}$ which is a multiple of a codeword in Γ. To solve the tracing problem we must construct a set $\Gamma \subseteq \mathbb{F}_q^{2k}$ 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 $\mathbb{F}_q^{2k}$. 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:

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

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.

Non-Black-Box tracing algorithm: Consider the set of vectors in Γ. Let $\bar{d} \in \mathbb{F}_q^{2k}$ 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 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 $\bar{v} - \bar{w} = \langle f(1), \dots, f(\ell) \rangle$.

Step 2: Since $\bar{w}$ has Hamming weight at most k, we know that $\langle f(1), \dots, f(\ell) \rangle$ 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 $\bar{v} \in \mathbb{F}_q^{\ell}$. Let g be a polynomial of degree at most k such that $g(i) = 0$ for all $i = 1, \dots, \ell$ for which $f(i) \neq v_i$ (where v_i is the i'th component of $\bar{v}$). Then we know that for all $i = 1, \dots, \ell$ 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 $i = 1, \dots, \ell$) 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:

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 [8] 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 $2k - 1$.

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

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 [13], runs in time $O(\ell \log \ell \log \log \ell)$.

Our scheme also supports two types of black box tracing techniques, which we describe in the next two sections.

4 Minimal Access Black box tracing against arbitrary pirates

In this section, we show a minimal access black box tracing algorithm for our basic encryption scheme that works against arbitrary pirates. To achieve this we introduce an easier tracing goal called *black box confirmation.*

4.1 Black Box Confirmation

When the police finds a pirate decoder they often have a suspect pirate in mind. The goal of black box confirmation is to allow the police to confirm their suspicion. A black box confirmation algorithm is defined as follows: the algorithm is given (1) the public key and all random bits used during initial key generation, (2) black box access to a pirate decoder $\mathcal{D}$, and (3) a set $T_{suspect}$ of private keys suspected of creating $\mathcal{D}$. The black box confirmation algorithm outputs either "`not-guilty`" or "`key` $\bar{d}$ `is guilty`" for some $\bar{d} \in T_{suspect}$. Let $T_{\mathcal{D}}$ be the set of private keys in the pirate's possession when $\mathcal{D}$ was created. The output must satisfy the following two requirements:

1. **Confirmation:** If $T_{\mathcal{D}} \subseteq T_{suspect}$ then the confirmation algorithm must pronounce at least one key $\bar{d} \in T_{suspect}$ as guilty.
2. **Soundness:** If the confirmation algorithm outputs `key` $\bar{d}$ `is guilty` then $\bar{d} \in T_{\mathcal{D}}$.

As always we assume that the size of both $T_{suspect}$ and $T_{\mathcal{D}}$ is less than the collusion bound k. The police does not know ahead of time whether $T_{\mathcal{D}} \subseteq T_{suspect}$. Hence, condition (2) ensures that when $T_{\mathcal{D}} \not\subseteq T_{suspect}$ the pirate cannot fool the algorithm into accusing an innocent user. Note that when the police has a small number of suspect sets $T^1_{suspect}, \ldots, T^r_{suspect}$ they can easily find a guilty key $\bar{d} \in T_{\mathcal{D}}$ using black box confirmation.

A black box confirmation algorithm will imply a black box tracing algorithm as follows. Run the confirmation algorithm on all $\binom{\ell}{k}$ candidate coalitions (ℓ is the total number of users in the system). By the confirmation property (Property 1) when the tested coalition contains the guilty coalition, some member of the guilty coalition will be pronounced guilty. A guilty user might be caught when other coalitions are tested. By the soundness property (property 2) whenever a key $\bar{d}$ is pronounced guilty the key is a member of the guilty coalition with very high probability. Thus we obtain a black box tracing algorithm with a running time of $O(\binom{\ell}{k} k^2)$. This is not an efficient tracing algorithm, but it shows that black box tracing is possible in principle. Hence, we do not have to rely on the pirate to embed specific information in the pirate decoder. In the next section we show an efficient black box tracing algorithm against a restricted set of pirates.

We briefly outline the basic idea behind our black box confirmation algorithm. Let $T_{suspect} = \{\bar{d}_1, \ldots, \bar{d}_k\}$ and let g be a generator of G_q. Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key. To confirm its suspicion of $T_{suspect}$ the tracer queries the pirate decoder with a random 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 $i \in T_{suspect}$. Here w is a random element of $\mathbb{F}_q$. This ciphertext is invalid since $\langle g^{z_1}, \ldots, g^{z_{2k}} \rangle$ is most likely not of the form $\langle h_1^r, \ldots, h_{2k}^r \rangle$ for any r. We show that when $T_{\mathcal{D}} \subseteq T_{suspect}$ the pirate decoder cannot distinguish this invalid ciphertext from a real one. Consequently, we show that it must respond with $A = S/g^{\bar{z}\cdot\bar{d}}$ where $\bar{d}$ is some representation of y in the convex hull of $\bar{d}_1, \ldots, \bar{d}_k$. Hence, when $T_{\mathcal{D}} \subseteq T_{suspect}$ the pirate decoder always responds with $A = S/g^w$. When $T_{\mathcal{D}} \cap T_{suspect} = \emptyset$ the decoder's view is independent of S/g^w. Hence, the tracer can tell if $T_{suspect}$ contains one of the pirate's keys. With a bit more work the tracer can then point to a specific $\bar{d} \in T_{suspect}$ that must be in the pirate's possession.

4.2 Definitions and Distributions

Before we present the black box confirmation algorithm we need to define a few terms. For a given suspect set $T_{suspect} = \{\bar{d}_1, \ldots, \bar{d}_k\} \subseteq \mathbb{F}_q^{2k}$ define the following terms:

- For $i = 0, \ldots, k$ define $T_i = \{\bar{d}_1, \ldots, \bar{d}_i\} \subseteq T_{suspect}$. T_0 is the empty set.
- For $i = 0, \ldots, k$, and $W \in G_q$ define $\mathsf{CT}_i(W)$ as the set:

  $$\mathsf{CT}_i(W) = \left\{ C = \langle S, H_1, \ldots, H_{2k} \rangle \;\middle|\; S \in G_q \text{ and } W = \prod_{i=1}^{2k} H_i^{\delta_i} \text{ for all } \bar{d} = \langle \delta_1, \ldots, \delta_{2k} \rangle \in T_i \right\}$$

  Define $\mathsf{CT}_i = \bigcup_{W \in G_q} \mathsf{CT}_i(W)$.
  Note that when key $\bar{d} \in T_i$ is used to decrypt $C \in \mathsf{CT}_i(W)$ we get S/W.
- For $i = 0, \ldots, k$ define the distribution CW_i on pairs $\langle W, C \rangle$ as follows: pick a random $W \in G_q$, pick a random $C \in \mathsf{CT}_i(W)$, output $\langle W, C \rangle$.
- For $i = 0, \ldots, k$ define $P_{\mathcal{D},i}$ as: $\quad P_{\mathcal{D},i} = \Pr_{\langle W,C \rangle \in \mathsf{CW}_i} [\mathcal{D}(C, S/W) = \mathtt{valid}]$.
  Where $C = \langle S, H_1, \ldots, H_{2k} \rangle$. Here $\langle W, C \rangle$ is chosen from the distribution CW_i and $\mathcal{D}(C, S/W)$ denotes the success or failure of the pirate decoder on input $C, S/W$ (minimal access).
- Given a public key $\langle y, h_1, \ldots, h_{2k} \rangle$ we refer to the set $\{\langle S, h_1^r, \ldots, h_{2k}^r \rangle : r \in \mathbb{F}_q\}$ as the set of valid ciphertexts. We refer to ciphertexts outside this set as invalid ciphertexts.
  Define the distribution CW_{valid} on pairs $\langle W, C \rangle$ as follows: (1) pick a random $r \in \mathbb{F}_q$ and $S \in G_q$, (2) set $W = y^r$, and $C = \langle S, h_1^r, \ldots, h_{2k}^r \rangle$, (3) output $\langle W, C \rangle$. Note that C is a valid ciphertext and its decryption is S/W.

Note that for any i we can efficiently sample from the distribution CW_i as follows: (1) pick a random $w \in \mathbb{F}_q$ and set $W = g^w$, (2) pick a random vector $\bar{z} = \langle z_1, \ldots, z_{2k} \rangle \in \mathbb{F}_q^{2k}$ such that $\bar{z} \cdot \bar{d} = w$ for all $\bar{d} \in T_i$, (3) pick a random $S \in G_q$, (4) set $C = \langle S, g^{z_1}, \ldots, g^{z_{2k}} \rangle$, and output $\langle W, C \rangle$.

We claim that $P_{\mathcal{D},0} \leq 1/q$. To see this observe that when $\langle W, C \rangle$ is chosen from CW_0 we have that W is independent of C. It follows that $\mathcal{D}(C, S/W)$ is valid with probability at most $1/q$.

4.3 Description of Black Box Confirmation Algorithm

We are now ready to describe the black box confirmation algorithm. An outline of the algorithm is given in [4]. The algorithm is given minimal access to $\mathcal{D}$ and works as follows:

Algorithm 1:

Step 1: Compute estimates for every $P_{\mathcal{D},i}$.

For each $i = 0, \ldots, k$ do:

Step 1a: For $j = 1, \ldots, \lambda$ pick random and independent pairs $\langle W_j, C_j \rangle$ chosen according to the distribution CW_i. The value of λ will be determined later.

Step 1b: Run $\mathcal{D}$ on all inputs $C_1, \ldots, C_\lambda$ and let c_i be the number of times that $\mathcal{D}(C_j, S_j/W_j) =$ `valid`.

Step 1c: Set $\hat{p}_i = c_i/\lambda$.

Step 2: If $\hat{p}_k \leq 3/4$ output `not-guilty` and stop.

Step 3: Otherwise, since $P_{\mathcal{D},0} \leq 1/q$ it follows that $\hat{p}_0 \leq 1/4$ with high probability (we calculate exact probabilities below). Then:

$$\frac{1}{2} \leq |\hat{p}_k - \hat{p}_0| = \left| \sum_{i=1}^{k} \hat{p}_i - \hat{p}_{i-1} \right| \leq \sum_{i=1}^{k} |\hat{p}_i - \hat{p}_{i-1}|$$

It follows that there exists a $1 \leq j \leq k$ such that: $|\hat{p}_j - \hat{p}_{j-1}| \geq \frac{1}{2k}$.

Output $\bar{d}_j \in T_{suspect}$ is guilty, and stop.

The completes the description of the black box confirmation algorithm. The algorithm requires $\lambda(k+1)$ queries to the pirate decoder $\mathcal{D}$. As we show in Lemma 4.6, we take λ on the order of $k \log k$ and hence the algorithm has running time $O(k^2 \log k)$. Note that Algorithm 1 does not need the public key or the private bits used during key generation, only the k private keys of the suspect coalition.

In the rest of this section, we will prove the following theorem.

Theorem 4.1. *Algorithm 1 is a black box confirmation algorithm.*

To prove Theorem 4.1, we must show that Algorithm 1 satisfies the confirmation property (Property 1) and the soundness property (Property 2) for black box confirmation. These will be proven in Lemma 1 and Lemma 4.6 respectively. The probability space in both lemmas is over the random bits used by Algorithm 1, the random bits used by the pirate and pirate decoder, and the random set of k private keys given to the pirate.

4.4 Algorithm 1 Satisfies the Confirmation Property

We show that Algorithm 1 satisfies the confirmation property, i.e., no pirate can fool Algorithm 1 when $T_{\mathcal{D}} \subseteq T_{suspect}$. To do so, we show that the following *non-traceable pirate* does not exist.

Non-traceable pirate: Let $\mathcal{P}$ be some pirate. Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key and let $T_{\mathcal{D}} = \{\bar{d}_1, \ldots, \bar{d}_k\}$ be a random set of k private keys. We say that the pirate $\mathcal{P}$ is ϵ-*non-traceable* if given the public key and $T_{\mathcal{D}}$, the pirate creates a pirate decoder $\mathcal{D}$ with the following properties: (1) $\mathcal{D}$ correctly decrypts all valid ciphertexts, and (2) when $T \subseteq T_{suspect}$ the pirate decoder $\mathcal{D}$ causes Algorithm 1 to output "`not-guilty`" with probability at least ϵ. The following lemma shows that a non-traceable pirate does not exist. Recall that u is the number of samples used in Step 1 of Algorithm 1.

Lemma 4.2. *Let $\epsilon > 0$ and let $\lambda = 1$. Suppose $\mathcal{P}$ is an ϵ-non-traceable pirate. Then $\mathcal{P}$ can be used to distinguish DH tuples from random tuples with advantage ϵ.*

Proof Suppose $T_{\mathcal{D}} \subseteq T_{suspect}$. Then, by assumption, when Algorithm 1 is given a decoder $\mathcal{D}$ created by $\mathcal{P}$, Algorithm 1 outputs `not-guilty` with probability at least ϵ. This means $\hat{p}_k \leq 3/4$ with probability at least ϵ. Since $\lambda = 1$ it follows that $P_{\mathcal{D},k} < 1 - \epsilon$. This means that the pirate $\mathcal{P}$ is able to build a pirate decoder $\mathcal{D}$ that correctly decrypts all valid ciphertexts; however for a random $C \in \mathsf{CT}_k(W)$ it outputs "invalid" on input $C, S/W$ with probability at least ϵ.

We show how the pirate $\mathcal{P}$ can be used to solve DDH in G_q. Given a challenge tuple $\langle g_1, g_2, u_1, u_2\rangle$ we use the following algorithm to decide whether it is a random tuple or a Diffie-Hellman tuple. We write $g_2 = g_1^c$ for some unknown c.

Algorithm $\mathcal{A}$:

Step 1: Choose a random set of k codewords in Γ. Denote these by $\gamma^{(1)}, \ldots, \gamma^{(k)} \in \mathbb{F}_q^{2k}$. (These will correspond to the k private keys given to the pirate.)

Step 2: Pick a random $A \in \mathbb{F}_q$ such that $g_1^A g_2 \neq 1$.

Step 3: Pick random vectors $\bar{a}, \bar{b} \in \mathbb{F}_q^{2k}$ such that the vector $\bar{a} - A\bar{b}$ is orthogonal to the k vectors $\gamma^{(1)}, \ldots, \gamma^{(k)}$. Write $\bar{a} = (a_1, \ldots, a_{2k})$ and $\bar{b} = (b_1, \ldots, b_{2k})$.

Step 4: Set $h_i = g_1^{a_i} g_2^{b_i}$ for $i = 1, \ldots, 2k$.

Step 5: Pick a random vector $\bar{\alpha} \in \mathbb{F}_q^{2k}$ such that $\bar{\alpha}$ is orthogonal to $\bar{a} - A\bar{b}$. Write $\bar{\alpha} = (\alpha_1, \ldots, \alpha_{2k})$.

Step 6: Set $y = h_1^{\alpha_1} \cdots h_{2k}^{\alpha_{2k}}$. Note that $y = g_1^{\bar{\alpha}\cdot\bar{a}} g_2^{\bar{\alpha}\cdot\bar{b}} = g_1^{A(\bar{\alpha}\cdot\bar{b})} g_2^{\bar{\alpha}\cdot\bar{b}} = (g_1^A g_2)^{\bar{\alpha}\cdot\bar{b}}$.

Step 7: For $i = 1, \ldots, k$ set $\theta_i = \dfrac{\bar{\alpha}\cdot\bar{b}}{\gamma^{(i)}\cdot\bar{b}}$. Set $\bar{d}_i = \theta_i \gamma^{(i)}$.

Observe that $\bar{d}_1, \ldots, \bar{d}_k$ are representations of y to the base $h_1, \ldots, h_{2k}$. Indeed, for $\bar{d}_i = (\delta_1, \ldots, \delta_{2k})$ we have:

$$\prod_{i=1}^{2k} h_i^{\delta_i} = g_1^{\bar{a}\cdot\bar{d}_i} g_2^{\bar{b}\cdot\bar{d}_i} = \left[g_1^{\bar{a}\cdot\gamma^{(i)}} g_2^{\bar{b}\cdot\gamma^{(i)}}\right]^{\theta_i} = \left[g_1^{A(\bar{b}\cdot\gamma^{(i)})} g_2^{(\bar{b}\cdot\gamma^{(i)})}\right]^{\theta_i} = \left(g_1^A g_2\right)^{\bar{b}\cdot\bar{\alpha}} = y$$

Step 8: Run the pirate $\mathcal{P}$ giving it the public key $\langle y, h_1, \ldots, h_{2k}\rangle$ and the k private keys $\theta_1, \ldots, \theta_k$. The pirate builds a pirate decoder $\mathcal{D}$.

Step 9: Pick a random $S \in G_q$ and construct the ciphertext: $\tilde{C} = \langle S,\ {u_1}^{a_1} {u_2}^{b_1},\ \ldots,\ {u_1}^{a_{2k}} u_2^{b_{2k}}\rangle$
Let $\tilde{C} = \langle S, H_1, \ldots, H_{2k}\rangle$ and set $W = \prod H_i^{\alpha_i}$

Step 10: Query the pirate decoder $\mathcal{D}$ on the ciphertext/message pair $\langle \tilde{C},\ S/W\rangle$. If $\mathcal{D}$ returns "`valid`" output that $\langle g_1, g_2, u_1, u_2\rangle$ is a Diffie-Hellman tuple. Otherwise, output that $\langle g_1, g_2, u_1, u_2\rangle$ is a random tuple.

This concludes the description of algorithm $\mathcal{A}$ for solving DDH. We show that algorithm $\mathcal{A}$ above correctly decides whether the given tuple $\langle g_1, g_2, u_1, u_2\rangle$ is a DDH tuple. This follows from the following three claims.

Claim 4.3. *The distribution on the public key $\langle y, h_1, \ldots, h_{2k}\rangle$ and the k private keys $\theta_1, \ldots, \theta_k$ given to the pirate $\mathcal{P}$ in Step 8 is identical to the distribution generated by the real key generation algorithm.*

Proof Consider the key generation algorithm of Section 3. Observe that given $h_1, \ldots, h_{2k}$ and y the values $\theta_1, \ldots, \theta_k$ are determined. Indeed, given the values $y, h_1, \ldots, h_{2k}$ the vector $r_1, \ldots, r_{2k} \in \mathbb{F}_q$ and $\sum_{i=1}^{2k} r_i \alpha_i$ are determined. Then the θ_i are determined from equation (1). Hence, when using the real key generation algorithm the probability of seeing a specific valid sequence $\langle y, h_1, \ldots, h_{2k}, \theta_1, \ldots, \theta_k\rangle$ is:

$$\Pr[\langle y, h_1, \ldots, h_{2k}, \theta_1, \ldots, \theta_k\rangle] = 1/q^{2k+1}$$

where the probability is over the random bits used by the key generation algorithm.

We show that the same holds for the public key given to the pirate in Step 8. Let Γ_0 be the $k \times 2k$ matrix whose rows are the vectors $\gamma^{(1)}, \ldots, \gamma^{(k)} \in \mathbb{F}_q^{2k}$. This matrix has rank k. We first show that given the k linear constraints on $\bar{a}, \bar{b}$ in Step 3, the vector $\langle h_1, \ldots h_{2k}\rangle$ is uniformly distributed in G_q^{2k}.

Since $h_i = g_1^{a_i} g_2^{b_i} = g_1^{a_i+cb_i}$ it suffices to argue that given the k constraints in Step 3 the $2k$ values $a_i + cb_i$ are uniformly and independently distributed in F_q^{2k}. To do so it suffices to prove that the set of $3k$ relations $(\bar{a} - A\bar{b}) \cdot \gamma^{(i)} = 0$ for $i = 1, \dots, k$ and $a_j + cb_j = 0$ for $j = 1, \dots, 2k$ are linearly independent. In other words, we need to show that the rows of the following $3k \times 4k$ matrix are linearly independent:

$$\Omega = \begin{pmatrix} \Gamma_0 & -A\Gamma_0 \\ I_{2k} & cI_{2k} \end{pmatrix}$$

Here I_{2k} denotes the $2k \times 2k$ unit matrix. One can immediately verify that whenever $c \neq -A$ the matrix Ω has rank $3k$ as required. We know that $c \neq -A$ since in Step 2 we ensure that $g_1^A g_2 \neq 1$. This shows that the vector $\langle h_1, \dots, h_{2k} \rangle$ is uniformly distributed in G_q^{2k}.

Next, we show that y is uniformly distributed in G_q and is independent of $\langle h_1, \dots, h_{2k} \rangle$. This is immediate since $y = (g_1^A g_2)^{(\bar{\alpha} \cdot \bar{b})}$ and $\bar{\alpha} \cdot \bar{b}$ is uniform in $\mathbb{F}_q$. To see that $\bar{\alpha} \cdot \bar{b}$ is uniform (over the choice of $\bar{\alpha}$) recall that in Step 5 $\bar{\alpha}$ is generated as a random vector in $\mathbb{F}_q^{2k}$ subject to one constraint $\bar{\alpha} \cdot (\bar{a} - A\bar{b}) = 0$. This constraint on $\bar{\alpha}$ is linearly independent of the constraint $\bar{\alpha} \cdot \bar{b} = 0$. Hence, since $\bar{\alpha} \cdot \bar{b}$ is independent of $\langle h_1, \dots, h_{2k} \rangle$, so is y.

Finally, we show that given $y, h_1, \dots, h_{2k}$ the value of $\theta_1, \dots, \theta_k$ is determined. Hence, revealing $\theta_1, \dots, \theta_k$ does not further constrain $\bar{a}, \bar{b}, \bar{\alpha}$. To see this observe that since $\theta_i \gamma^{(i)}$ is a representation of y to the base $h_1, \dots, h_{2k}$ we have that:

$$\theta_i = \frac{(\bar{a} + c\bar{b}) \cdot \bar{\alpha}}{(\bar{a} + c\bar{b}) \cdot \gamma^{(i)}} = \frac{(A + c) \cdot (\bar{b} \cdot \bar{\alpha})}{(\bar{a} + c\bar{b}) \cdot \gamma^{(i)}}$$

But $(A + c)(\bar{b} \cdot \bar{\alpha})$ is completely determined by y, and the vector $\bar{a} + c\bar{b}$ is completely determined by $h_1, \dots, h_{2k}$. Thus, given $y, h_1, \dots, h_{2k}$ the value of $\theta_1, \dots, \theta_k$ is fixed.

To summarize, the probability that in Step 8 the pirate is given the input $\langle y, h_1, \dots, h_{2k}, \theta_1, \dots, \theta_k \rangle$ is $1/q^{2k+1}$ as required. Overall, we have shown that once $\langle y, h_1, \dots, h_{2k}, \theta_1, \dots, \theta_k \rangle$ is revealed there are $3k$ relations induced on $\bar{a}, \bar{b}$ and two relations induced on $\bar{\alpha}$ (one by y the other by Step 5). As a result, $\langle \bar{a}, \bar{b} \rangle$ is random in a linear space of dimension k and $\bar{\alpha}$ is random in a linear space of dimension $2k - 2$. This will be used in the next two claims. □

Claim 4.4. *Suppose $\langle g_1, g_2, u_1, u_2 \rangle$ is a random Diffie-Hellman tuple. Then given the public key $\langle y, h_1, \dots, h_{2k} \rangle$ and the k private keys $\theta_1, \dots, \theta_k$ the pair $\langle W, \tilde{C} \rangle$ generated in Step 9 is distributed according to CW_{valid}.*

Proof When $\langle g_1, g_2, u_1, u_2 \rangle$ is a Diffie-Hellman tuple we have that $u_1 = g_1^x$ and $u_2 = g_2^x$ for some random unknown x. It follows that $\tilde{C}$ generated in Step 9 satisfies:

$$\tilde{C} = \langle S, (g_1^{a_1} g_2^{b_1})^x, \dots, (g_1^{a_{2k}} g_2^{b_{2k}})^x \rangle = \langle S, h_1^x, \dots, h_{2k}^x \rangle$$

Since x and S are random and independent of $\langle y, h_1, \dots, h_{2k} \rangle$ this is a random valid ciphertext. Furthermore $W = u_1^{\bar{a} \cdot \bar{\alpha}} u_2^{\bar{b} \cdot \bar{\alpha}} = (g_1^A g_2)^{x(\bar{b} \cdot \bar{\alpha})} = y^x$ as required. □

Claim 4.5. *Suppose $\langle g_1, g_2, u_1, u_2 \rangle$ is a random tuple. Then given the public key $\langle y, h_1, \dots, h_{2k} \rangle$ and the k private keys $\theta_1, \dots, \theta_k$ the pair $\langle W, \tilde{C} \rangle$ generated in Step 9 is distributed according to CW_k.*

Proof We write $g_2 = g_1^c$ for some unknown c. We also let $u_1 = g_1^{x_1}$ and $u_2 = g_2^{x_2}$, and assume that $x_1 \neq x_2$. We need to show that the pair $\langle W, \tilde{C} \rangle$ generated in Step 9 satisfies: (1) W is uniform in G_q and independent of $I = \langle y, h_1, \dots, h_{2k}, \theta_1, \dots, \theta_k \rangle$, and (2) $\tilde{C}$ is uniform in $\mathsf{CT}_k(W)$ and independent of $\langle I, W \rangle$.

We start with W. Define $w = (Ax_1 + cx_2)(\bar{\alpha} \cdot \bar{b})$. Then:

$$W = \prod_{i=1}^{2k} (u_1^{a_i} u_2^{b_i})^{\alpha_i} = u_1^{\bar{a}\cdot\alpha} u_2^{\bar{b}\cdot\alpha} = u_1^{A\bar{b}\cdot\alpha} u_2^{\bar{b}\cdot\alpha} = g_1^{(Ax_1+cx_2)(\bar{b}\cdot\alpha)} = g^w$$

To see that given I the value W is uniform in G_q it suffices to show that w is uniform in $\mathbb{F}_q$. This is immediate since $\bar{\alpha} \cdot \bar{b} \neq 0$ (since $y \neq 1$) and x_1, x_2 are independent of I.

Next, we show that given I, W the ciphertext $\tilde{C}$ is uniform in $\mathsf{CT}_k(W)$. First, observe that $\tilde{C}$ generated in Step 9 satisfies $\tilde{C} \in \mathsf{CT}_k(W)$. To see this write $\tilde{C} = \langle S, H_1, \dots, H_{2k} \rangle$. Then $H_i = u_1^{a_i} u_2^{b_i}$, and thus for any $\bar{d_i} = \theta_i \gamma^{(i)} = (\delta_1, \dots, \delta_{2k})$ we have:

$$\prod_{i=1}^{2k} H_i^{\delta_i} = u_1^{\bar{a}\cdot\bar{d}_i} u_2^{\bar{b}\cdot\bar{d}_i} = \left[g_1^{x_1(\bar{a}\cdot\gamma^{(i)})} g_2^{x_2(\bar{b}\cdot\gamma^{(i)})} \right]^{\theta_i} = \left[g_1^{x_1 A} g_2^{x_2} \right]^{\theta_i(\bar{b}\cdot\gamma^{(i)})} = \left[g_1^{x_1 A} g_2^{x_2} \right]^{\bar{b}\cdot\bar{\alpha}} = g_1^w = W$$

Therefore $\tilde{C} \in \mathsf{CT}_k(W)$. Next, we show that given I, W the vector $\tilde{C}$ is uniform in $\mathsf{CT}_k(W)$ over the choice of $\bar{a}, \bar{b}$. To do so we show that if $\tilde{C} = \langle S, g_1^{z_1}, \dots, g_1^{z_{2k}} \rangle$ then $\bar{z} = \langle z_1, \dots, z_{2k} \rangle$ is uniform in a linear space of dimension k. Since $\mathsf{CT}_k(W)$ is associated to a linear space of dimension k and $\tilde{C} \in \mathsf{CT}_k(W)$ it will follow that $\tilde{C}$ spans all of $\mathsf{CT}_k(W)$ over the choice of $\bar{a}, \bar{b}$ and is thus uniform in it.

Let $\tilde{C} = \langle S, g_1^{z_1}, \dots, g_1^{z_{2k}} \rangle$. Then by definition of $\tilde{C}$ we know that $\bar{z} = x_1 \bar{a} + cx_2 \bar{b}$. Therefore, it suffices to show that given I, W the vector $x_1 \bar{a} + cx_2 \bar{b}$ spans a space of dimension k over the choice of $\bar{a}, \bar{b}$. In Claim 4.3 we showed that $I = \langle y, h_1, \dots, h_{2k}, \theta_1, \dots, \theta_k \rangle$ induces $3k$ linear relations on the pair $\langle \bar{a}, \bar{b} \rangle$. Let Ω be the matrix as in Claim 4.3. We need to show that given that $\Omega \cdot [\bar{a} \| \bar{b}] = \bar{t}$, for some fixed $\bar{t} \in \mathbb{F}_q^{3k}$, the vector $x_1 \bar{a} + cx_2 \bar{b}$ spans a space of dimension k over the choice of $\bar{a}, \bar{b}$. This amounts to showing that the following $5k \times 4k$ matrix has full rank (i.e. has rank 4k):

$$\begin{pmatrix} \Gamma_0 & -A\Gamma_0 \\ I_{2k} & cI_{2k} \\ x_1 I_{2k} & cx_2 I_{2k} \end{pmatrix}$$

This is immediate since when $x_1 \neq x_2$ the two bottom parts of the matrix already have rank $4k$. This shows that given I, W the ciphertext $\tilde{C}$ comes from a linear space of dimension k. Hence, $\tilde{C}$ is uniform in $\mathsf{CT}_k(W)$ over the choice of $\bar{a}, \bar{b}$. This concludes the proof that given I the pair $\langle W, \tilde{C} \rangle$ is distributed according to CW_k. $\square$

Proof of Lemma 4.2: Claim 4.3 shows that in Step 8 of Algorithm $\mathcal{A}$ the pirate $\mathcal{P}$ is given input from a distribution that is identical to the distribution in the real attack. Consequently, it generates a pirate decoder $\mathcal{D}$ with the following properties. Let $\tilde{C} = \langle S, H_1, \dots, H_{2k} \rangle$ for some $H_1, \dots, H_{2k}$, let $W = \prod_{i=1}^{2k} H_i^{\alpha_i}$, and let $M = S/W$. When $\tilde{C}$ is a valid ciphertext then, by definition, M is the decryption of $\tilde{C}$. Given the pair $\langle \tilde{C}, M \rangle$ the decoder $\mathcal{D}$ satisfies: (1) if $\langle W, \tilde{C} \rangle$ is distributed according to CW_{valid} then the decoder returns "`valid`", and (2) if $\langle W, \tilde{C} \rangle$ is distributed according to CW_k it returns "`invalid`" with probability at least ϵ. By Claims 4.4 and 4.5 the pair $\langle W, \tilde{C} \rangle$ constructed in Step 9 falls into cases (1) or (2) depending on whether the given tuple $\langle g_1, g_2, u_1, u_2 \rangle$ is a Diffie-Hellman tuple. Hence, when the algorithm outputs its decision in Step 10 we have that:

$$\left| \Pr_{a,b \in \mathbb{F}_q} [\mathcal{A}(g, g^a, g^b, g^{ab}) = \text{"yes"}] - \Pr_{a,b,c \in \mathbb{F}_q} [\mathcal{A}(g, g^a, g^b, g^c) = \text{"yes"}] \right| \geq \epsilon$$

This concludes the proof of Lemma 4.2. $\square$

4.5 Algorithm 1 Satisfies the Soundness Property

The next lemma shows that Algorithm 1 satisfies the soundness property (property 2). Recall that λ is the number of samples we make in Step 1 of Algorithm 1.

Lemma 4.6. *Let $\mathcal{D}$ be a pirate decoder built using a set $T_{\mathcal{D}}$ of at most k private keys. Let $T_{suspect}$ be a suspect set of size at most k. Let $\epsilon > 0$ and set $\lambda = 64k \log \frac{k}{\epsilon}$. Then when Algorithm 1 interacts with $\mathcal{D}$ and outputs "`key` $\bar{d}$ `is guilty`" then $\bar{d} \in T_{\mathcal{D}}$ with probability at least $1-\epsilon$.*

Proof Since $\lambda = 64k \log \frac{k}{\epsilon}$ we know, by the Chernoff bound, that for all $i = 0, \ldots, k$ we have $|\hat{p}_i - P_{\mathcal{D},i}| < 1/8k$ with probability at least $1-\epsilon$. Therefore, if key $\bar{d}_j \in T_{suspect}$ is pronounced guilty then with probability at least $1-\epsilon$ we have

$$|P_{\mathcal{D},j} - P_{\mathcal{D},j-1}| \geq \frac{1}{4k}$$

This means that the pirate decoder is able to distinguish between the distributions CW_j and CW_{j-1} with advantage at least $\frac{1}{4k}$. The following lemma shows that if $\bar{d}_j \notin T_{\mathcal{D}}$ then this is not possible unless DDH in G_q cannot be solved with advantage $1/4k^2$. This will immediately prove Lemma 4.6

Lemma 4.7. *Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key. Let $T_{suspect} = \{\bar{d}_1, \ldots, \bar{d}_k\}$ be the set of k private keys used to define the sets CT_i for $i = 0, \ldots, k$. Suppose $\mathcal{P}$ is a pirate that given the public key and a random set of k private keys $T_{\mathcal{D}}$ is able to construct a decoder $\mathcal{D}$ with the following property:*

$$\exists j_0 \in \{1, \ldots, k\}: \quad \bar{d}_{j_0} \notin T_{\mathcal{D}} \quad \textit{and} \quad |P_{\mathcal{D},j_0} - P_{\mathcal{D},j_0-1}| \geq \epsilon$$

Then the pirate $\mathcal{P}$ can be used to solve DDH in G_q with advantage at least ϵ/k.

Proof Since $|P_{\mathcal{D},j_0} - P_{\mathcal{D},j_0-1}| \geq \epsilon$ we know that the pirate is able to build a decoder $\mathcal{D}$ that can distinguish CW_{j_0} from CW_{j_0-1} with advantage ϵ, without knowing $\bar{d}_j$. We show how to use such a pirate to solve DDH in G_q with advantage ϵ/k. Given a challenge tuple $\langle g_1, g_2, u_1, u_2 \rangle$ we use the following algorithm to decide whether it is a random tuple or a Diffie-Hellman tuple. We write $g_2 = g_1^c$ for some unknown c.

Algorithm $\mathcal{B}$:

Step 0: Pick a random $j_0 \in \{1, \ldots, k\}$.

Step 1: Choose a random set of k codewords in Γ. Denote these by $\gamma^{(1)}, \ldots, \gamma^{(k)} \in \mathbb{F}_q^{2k}$. (These will correspond to the k private keys given to the pirate.) Also, let $\gamma_s^{(1)}, \ldots, \gamma_s^{(k)}$ be an arbitrary set of k codewords in Γ. These will correspond to keys that belong to the suspect coalition. We assume $\gamma_s^{(j_0)} \notin \{\gamma^{(1)}, \ldots, \gamma^{(k)}\}$.

Step 2: Pick a random $A \in \mathbb{F}_q$ such that $g_1^A g_2 \neq 1$.

Step 3: Pick random vectors $\bar{a}, \bar{b} \in \mathbb{F}_q^{2k}$ such that the vector $\bar{a} - A\bar{b}$ is orthogonal to all vectors $\langle \gamma^{(1)}, \ldots, \gamma^{(k)}, \gamma_s^{(1)}, \ldots, \gamma_s^{(j_0-1)} \rangle$. If the chosen $\bar{a}, \bar{b}$ satisfies that $\bar{a} - A\bar{b}$ is orthogonal to $\gamma_s^{(j_0)}$ then repeat Step 3. Write $\bar{a} = (a_1, \ldots, a_{2k})$ and $\bar{b} = (b_1, \ldots, b_{2k})$.

Step 4: Set $h_i = g_1^{a_i} g_2^{b_i}$ for $i = 1, \ldots, 2k$.
Pick a random vector $\bar{v} \in \mathbb{F}_q^{2k}$ such that $\bar{v}$ is orthogonal to all $\gamma_s^{(1)}, \ldots, \gamma_s^{(j_0)}$.
Let $\bar{v} = \langle v_1, \ldots, v_{2k} \rangle$.

Step 5: Pick a random vector $\bar{\alpha} \in \mathbb{F}_q^{2k}$ such that $\bar{\alpha}$ is orthogonal to $\bar{a} - A\bar{b}$ and is orthogonal to $\bar{v}$. Write $\bar{\alpha} = (\alpha_1, \ldots, \alpha_{2k})$.

Step 6: Set $y = h_1^{\alpha_1} \cdots h_{2k}^{\alpha_{2k}}$. Then $y = g_1^{\bar{\alpha}\cdot\bar{a}} g_2^{\bar{\alpha}\cdot\bar{b}} = g_1^{A(\bar{\alpha}\cdot\bar{b})} g_2^{\bar{\alpha}\cdot\bar{b}} = (g_1^A g_2)^{\bar{\alpha}\cdot\bar{b}}$.

Step 7: For $i = 1, \ldots, k$ set $\theta_i = \dfrac{\bar{\alpha} \cdot \bar{b}}{\gamma^{(i)} \cdot \bar{b}}$. Set $\bar{d}_i = \theta_i \gamma^{(i)}$ and set $T_{\mathcal{D}} = \{\bar{d}_1, \ldots, \bar{d}_k\}$.

For $i = 1, \ldots, j_0 - 1$ set $\theta_i^s = \dfrac{\bar{\alpha} \cdot \bar{b}}{\gamma_s^{(i)} \cdot \bar{b}}$. Set $\bar{d}_i^s = \theta_i^s \gamma_s^{(i)}$.

As in Algorithm $\mathcal{A}$ we know that $\bar{d}_1, \ldots, \bar{d}_k$ and $\bar{d}_1^s, \ldots, \bar{d}_{j_0-1}^s$ are representations of y to the base $h_1, \ldots, h_{2k}$.

Step 8: Run the pirate $\mathcal{P}$ giving it the public key $\langle y, h_1, \ldots, h_{2k} \rangle$ and the k private keys $\theta_1, \ldots, \theta_k$. The pirate builds a pirate decoder $\mathcal{D}$.

Step 9: Pick a random $S \in G_q$ and construct the ciphertext: $\tilde{C} = \langle S,\ g_1^{v_1} {u_1}^{a_1} {u_2}^{b_1},\ \ldots,\ g_1^{v_{2k}} {u_1}^{a_{2k}} u_2^{b_{2k}} \rangle$
Let $\tilde{C} = \langle S, H_1, \ldots, H_{2k} \rangle$ and set $W = \prod_{i=1}^{2k} H_i^{\alpha_i}$.

Step 10: Query the pirate decoder $\mathcal{D}$ on the ciphertext/message pair $\langle \tilde{C},\ S/W \rangle$. If $\mathcal{D}$ returns "`valid`" output that $\langle g_1, g_2, u_1, u_2 \rangle$ is a Diffie-Hellman tuple. Otherwise, output that $\langle g_1, g_2, u_1, u_2 \rangle$ is a random tuple.

This concludes the description of algorithm $\mathcal{B}$ for solving DDH. We show that algorithm $\mathcal{B}$ above correctly decides whether the given tuple $\langle g_1, g_2, u_1, u_2 \rangle$ is a DDH tuple. This follows from the following three claims.

Claim 4.8. *The distribution on the public key $\langle y, h_1, \ldots, h_{2k} \rangle$ and the k private keys $\theta_1, \ldots, \theta_k$ given to the pirate $\mathcal{P}$ in Step 8 is identical to the distribution generated by the real key generation algorithm.*

Proof The proof is identical to the proof of Claim 4.3 with minor changes. The only difference is that the pair $\langle \bar{a}, \bar{b} \rangle$ is subject to $k + j_0 - 1$ constraints in Step 3. Hence, given the public key $\langle \bar{a}, \bar{b} \rangle$ sits in a space of dimension $k - j_0 + 1$. □

Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be the public key generated in Step 6. For $i = 1, \ldots, k$ define $\theta_i^s \in \mathbb{F}_p$ as the value that makes $\theta_i^s \gamma_s^{(i)}$ be a representation of y base $h_1, \ldots, h_{2k}$. For $i = 1, \ldots, j_0 - 1$ the value θ_i^s is computed in Step 7, but for $i = j_0, \ldots, k$ the value θ_i^s is unknown to Algorithm $\mathcal{B}$. For $i = 1, \ldots, k$ define $\bar{d}_i^s = \theta_i^s \gamma_s^{(i)}$. As above, $\bar{d}_i^s$ is unknown to Algorithm $\mathcal{B}$ for $i = j_0, \ldots, j$.

Claim 4.9. *Let $T_{suspect} = \{\bar{d}_1^s, \ldots, \bar{d}_k^s\}$ be the suspect set used to define the sets $\mathsf{CT}_0, \ldots, \mathsf{CT}_k$. Suppose $\langle g_1, g_2, u_1, u_2 \rangle$ is a random Diffie-Hellman tuple. As in Step 1 we assume $\bar{d}_{j_0}^s \notin T_{\mathcal{D}}$. Then given the public key $\langle y, h_1, \ldots, h_{2k} \rangle$ and the k private keys $\theta_1, \ldots, \theta_k$ the pair $\langle W, \tilde{C} \rangle$ generated in Step 9 is distributed according to CW_{j_0}.*

Proof We write $g_2 = g_1^c$ for some unknown c. We also let $u_1 = g_1^x$ and $u_2 = g_2^x$ for some unknown x. We need to show that the pair $\langle W, \tilde{C} \rangle$ generated in Step 9 satisfies: (1) W is uniform in G_q and independent of $I = \langle y, h_1, \ldots, h_{2k}, \theta_1, \ldots, \theta_k \rangle$, and (2) $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0}(W)$ and independent of $\langle I, W \rangle$.

We start with W. Define $w = x(A + c)(\bar{b} \cdot \bar{\alpha})$. Then, since $\bar{\alpha}$ is orthogonal to $\bar{v}$ we have:

$$W = \prod_{i=1}^{2k} (g_1^{v_i} u_1^{a_i} u_2^{b_i})^{\alpha_i} = g_1^{\bar{v} \cdot \bar{\alpha}} u_1^{\bar{a} \cdot \bar{\alpha}} u_2^{\bar{b} \cdot \bar{\alpha}} = u_1^{A(\bar{b} \cdot \bar{\alpha})} u_2^{\bar{b} \cdot \bar{\alpha}} = g_1^{x(A+c)(\bar{b} \cdot \bar{\alpha})} = g_1^w$$

To see that given I the value W is uniform in G_q it suffices to show that w is uniform in $\mathbb{F}_q$. This is immediate since $\bar{\alpha} \cdot \bar{b} \neq 0$ (since $y \neq 1$), and $A + c \neq 0$ (since $g_1^A g_2 \neq 1$), and x is independent of I.

Next, we show that given I, W the ciphertext $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0}(W)$. First, observe that $\tilde{C}$ generated in Step 9 satisfies $\tilde{C} \in \mathsf{CT}_{j_0}(w)$. To see this write $\tilde{C} = \langle S, H_1, \ldots, H_{2k} \rangle$. Then $H_i = g_1^{v_i} u_1^{a_i} u_2^{b_i}$. Recall that $\bar{v}$ is chosen so that it is orthogonal to any $\bar{d}_i^s$ for $i = 1, \ldots, j_0$. Therefore, for any $i = 1, \ldots, j_0$ and $\bar{d}_i^s = (\delta_1, \ldots, \delta_{2k})$ we have:

$$\prod_{i=1}^{2k} H_i^{\delta_i} = g_1^{\bar{v}\cdot\bar{d}_i^s} u_1^{\bar{a}\cdot\bar{d}_i^s} u_2^{\bar{b}\cdot\bar{d}_i^s} = \left[g_1^{(\bar{a}\cdot\gamma_s^{(i)})} g_2^{(\bar{b}\cdot\gamma_s^{(i)})}\right]^{x\theta_i^s} = \left[g_1^{(\bar{a}+c\bar{b})\cdot\gamma_s^{(i)}}\right]^{x\theta_i^s} = \left[g_1^{(\bar{a}+c\bar{b})\cdot\bar{\alpha}}\right]^{x} = \left[g_1^{(A+c)(\bar{b}\cdot\bar{\alpha})}\right]^{x} = g_1^w = W$$

Therefore $\tilde{C} \in \mathsf{CT}_{j_0}(W)$. Next, we show that given I, W the vector $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0}(W)$ over the choice of $\bar{v}$.

We first show that $\bar{v}$ is independent of I. Clearly $\bar{v}$ is independent of $h_1, \ldots, h_{2k}$. To show that it is independent of y we show that $\bar{v}$ is independent of $\bar{b}\cdot\bar{\alpha}$. That is,

$$\Pr_{\bar{v},\bar{\alpha}}\left[\bar{v} = \bar{v}_0 \,\middle|\, \bar{\alpha}\cdot\bar{v}_0 = 0,\ \bar{\alpha}\cdot\bar{b} = t,\ \bar{\alpha}\cdot(\bar{a} - A\bar{b}) = 0\right] = \Pr_{\bar{v},\bar{\alpha}}[\bar{v} = \bar{v}_0]$$

For any $\bar{v}_0$ that is orthogonal to all $\gamma_s^{(1)}, \ldots, \gamma_s^{(j_0)}$ and $t \in \mathbb{F}_q$. This follows from the fact that $\bar{v}_0$ is linearly independent of $\bar{b}$ since $\bar{b}\cdot\gamma_s^{(1)} \neq 0$, and $\bar{v}_0$ is linearly independent of $\bar{a} - A\bar{b}$ since since $\bar{a} - A\bar{b}$ is not orthogonal to $\gamma_s^{(j_0)}$

Since $\bar{v}$ is independent of I, W, and $\bar{v}$ is uniform in a space of dimension $2k - j_0$, we know that $\tilde{C}$ is associated to a linear space of dimension $2k - j_0$ and is independent of I, W. But $2k - j_0$ is also the dimension of the linear space associated to $\mathsf{CT}_{j_0}(W)$. Hence, given I, W the ciphertext $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0}(W)$. $\square$

Claim 4.10. *Let $T_{suspect} = \{\bar{d}_1^s, \ldots, \bar{d}_k^s\}$ be the suspect set used to define the sets $\mathsf{CT}_0, \ldots, \mathsf{CT}_k$. Suppose $\langle g_1, g_2, u_1, u_2\rangle$ is a random tuple. As in Step 1 we assume $\bar{d}_{j_0}^s \notin T_{\mathcal{D}}$. Then given the public key $\langle y, h_1, \ldots, h_{2k}\rangle$ and the k private keys $\theta_1, \ldots, \theta_k$ the pair $\langle W, \tilde{C}\rangle$ generated in Step 9 is distributed according to CW_{j_0-1}.*

Proof We write $g_2 = g_1^c$ for some unknown c. We also let $u_1 = g_1^{x_1}$ and $u_2 = g_2^{x_2}$ for some unknown x_1, x_2. We assume $x_1 \neq x_2$. We need to show that the pair $\langle W, \tilde{C}\rangle$ generated in Step 9 satisfies: (1) W is uniform in G_q and independent of $I = \langle y, h_1, \ldots, h_{2k}, \theta_1, \ldots, \theta_k\rangle$, and (2) $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0-1}(W)$ and independent of $\langle I, W\rangle$.

We start with W. Define $w = (x_1 A + c x_2)(\bar{\alpha}\cdot\bar{b})$. Then since $\bar{v}$ is orthogonal to $\bar{\alpha}$:

$$W = g_1^{\bar{v}\cdot\bar{\alpha}} u_1^{\bar{a}\cdot\bar{\alpha}} u_2^{\bar{b}\cdot\bar{\alpha}} = g_1^{x_1(\bar{a}\cdot\bar{\alpha})} g_2^{x_2(\bar{b}\cdot\bar{\alpha})} = \left[g_1^{x_1 A} g_2^{x_2}\right]^{(\bar{b}\cdot\bar{\alpha})} = g_1^w$$

To see that given I the value W is uniform in G_q it suffices to show that w is uniform in $\mathbb{F}_q$. This is immediate since $\bar{\alpha}\cdot\bar{b} \neq 0$ (since $y \neq 1$), and $Ax_1 + cx_2$ is independent of I.

Next, we show that given I, W the ciphertext $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0-1}(W)$. First, observe that $\tilde{C}$ generated in Step 9 satisfies $\tilde{C} \in \mathsf{CT}_{j_0-1}(W)$. To see this write $\tilde{C} = \langle S, H_1, \ldots, H_{2k}\rangle$. Then $H_i = g_1^{v_i} u_1^{a_i} u_2^{b_i}$. Recall that $\bar{v}$ is chosen so that it is orthogonal to any $\bar{d}_i^s$ for $i = 1, \ldots, j_0$. Therefore, for any $i = 1, \ldots, j_0 - 1$ and $\bar{d}_i^s = (\delta_1, \ldots, \delta_{2k})$ we have:

$$\prod_{i=1}^{2k} H_i^{\delta_i^s} = g_1^{\bar{v}\cdot\bar{d}_i^s} u_1^{\bar{a}\cdot\bar{d}_i^s} u_2^{\bar{b}\cdot\bar{d}_i^s} = \left[g_1^{x_1(\bar{a}\cdot\gamma_s^{(i)})} g_2^{x_2(\bar{b}\cdot\gamma_s^{(i)})}\right]^{\theta_i^s} = \left[g_1^{x_1 A} g_2^{x_2}\right]^{\theta_i^s(\bar{b}\cdot\gamma_s^{(i)})} = \left[g_1^{x_1 A} g_2^{x_2}\right]^{\bar{b}\cdot\bar{\alpha}} = g_1^w = W$$

Therefore $\tilde{C} \in \mathsf{CT}_{j_0-1}(W)$. Next, we show that given I, W the vector $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0-1}(W)$. When $\tilde{C} = \langle S, g_1^{z_1}, \ldots, g_1^{z_{2k}}\rangle$ we know that $\bar{z} = \bar{v} + x_1\bar{a} + cx_2\bar{b}$. Hence, it suffices to show that given

I, W the vector $\bar{z}$ is uniform in a space of dimension $2k - j_0 + 1$ over the choice of $\bar{v}$ and x_1, x_2. As in Claim 4.9 we know that $\bar{v}$ is independent of I, W, and therefore it is uniform in a space of dimension $2k - j_0$. Hence, it suffices to show that given I, W the vector $x_1\bar{a} + cx_2\bar{b}$ is uniform in a space of dimension 1 over the choice of x_1, x_2 and it is linearly independent of the space of dimension $2k - j_0$ spanned by $\bar{v}$. First, since $x_1 \neq x_2$ we know that $x_1\bar{a} + cx_2\bar{b}$ is independent of $h_1, \ldots, h_{2k}$. However, the value w induces one affine constraint on x_1, x_2 since $x_1 A + cx_2 = w/(\bar{\alpha} \cdot \bar{b})$. Hence, given I, W the vector $x_1\bar{a} + cx_2\bar{b}$ spans an affine space of dimension 1 as required. Note that when $x_1 A + cx_2 = 0$ the space $x_1\bar{a} + cx_2\bar{b}$ is a linear space of dimension 1. We show that this space is not contained in the space spanned by $\bar{v}$ by showing that when $Ax_1 + cx_2 = 0$ the vector $x_1\bar{a} + cx_2\bar{b}$ cannot be orthogonal to $\gamma_s^{(j_0)}$:

$$(x_1\bar{a} + cx_2\bar{b}) \cdot \gamma_s^{(j_0)} = x_1(\bar{a} - A\bar{b}) \cdot \gamma_s^{(j_0)} + (cx_2 + Ax_1)(\bar{b} \cdot \gamma_s^{(j_0)}) = x_1(\bar{a} - A\bar{b}) \cdot \gamma_s^{(j_0)}$$

But since in Step 3 we make sure that $\bar{a} - A\bar{b}$ is not orthogonal to $\gamma_s^{(j_0)}$ we obtain the desired result. Hence, given I, W the ciphertext $\tilde{C}$ is uniform in $\mathsf{CT}_{j_0-1}(W)$. This shows that given I the pair $\langle W, \tilde{C}\rangle$ is chosen according to the distribution CW_{j_0-1}. □

Proof of Lemma 4.7: By Claim 4.8 the pirate is given input sampled from a distribution that is identical to the distribution in the real attack. Hence, it constructs a pirate decoder $\mathcal{D}$ satisfying the conditions of Lemma 4.7 for some $j_0 \in \{1, \ldots, k\}$. Then Claims 4.9 and 4.10 show that when j_0 is guessed correctly in Step 0 the pair $\langle W, \tilde{C}\rangle$ generated in Step 9 is distributed according to CW_{j_0} or CW_{j_0-1} depending on whether $\langle g_1, g_2, u_1, u_2\rangle$ is a Diffie-Hellman tuple or a random tuple. Hence, when j_0 is guesses correctly, Algorithm $\mathcal{B}$ solves DDH in G_q with advantage ϵ. The probability that j_0 is guessed correctly is $1/k$, hence, Algorithm $\mathcal{B}$ solves DDH in G_q with advantage ϵ/k. □

5 Full Access Black box tracing against single-key pirates

Suppose a pirate obtains the private keys $\bar{d}_1, \ldots, \bar{d}_k$. A natural strategy for the pirate to build a pirate decryption box is to form a new random 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 show an efficient full access 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 her she is essentially acting as a single-key pirate.

Formally, we model the single-key pirate as two distinct parties $(\mathcal{P}_1, \mathcal{P}_2)$. The first party $\mathcal{P}_1$, called the *key-builder*, is given the public key, and k random private keys $\bar{d}_1, \ldots, \bar{d}_k$. It creates a new key $\bar{d}$ by forming a random convex combination of the given private keys. The key-builder then hands $\bar{d}$ to the second party $\mathcal{P}_2$, called the *box-builder*. The box-builder, seeing only $\bar{d}$ and the public key, is free to implement the pirate decryption box however it wants.

We show how the tracer can extract the representation $\bar{d} = (\delta_1, \ldots, \delta_{2k})$ from a pirate decoder created by a single-key pirate. Once $\bar{d}$ is obtained we use the algorithm of the Section 3.3 to recover the pirate keys. The basic idea in extracting $\bar{d}$ 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 decoder cannot distinguish invalid ciphertexts from valid ciphertexts (assuming the difficulty of DDH in G_q).

Hence, we show that on input $\tilde{C} = \langle S, H_1, \ldots, H_{2k} \rangle$ it must respond with A where

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

Non-traceable pirates: Let $(\mathcal{P}_1, \mathcal{P}_2)$ be a single-key pirate. Let $\langle y, h_1, \ldots, h_{2k} \rangle$ be a public key and let $\bar{d} = \{\delta_1, \ldots, \delta_{2k}\}$ be the private key generated by the key-builder $\mathcal{P}_1$ given the public key and a random set of k private keys. By assumption, $\bar{d}$ is a random convex combination of the k private keys. We say that the single-key pirate $(\mathcal{P}_1, \mathcal{P}_2)$ is *non-traceable* if given $\bar{d}$ the box builder $\mathcal{P}_2$ is able to create a pirate decoder $\mathcal{D}$ with the following properties: (1) $\mathcal{D}$ correctly decrypts all valid ciphertexts, and (2) when $\mathcal{D}$ is given a random invalid ciphertext $\tilde{C} = \langle S, H_1, \ldots, H_{2k} \rangle \in G_q^{2k+1}$ it outputs a value different from $S / \prod H_i^{\delta_i}$, with non-negligible probability (over the choice of $\tilde{C}$). The following lemma shows that a non-traceable single-key pirate does not exist.

Lemma 5.1. *Suppose $(\mathcal{P}_1, \mathcal{P}_2)$ is a non-traceable single-key pirate. Then $(\mathcal{P}_1, \mathcal{P}_2)$ can be used to solve DDH in G_q.*

Proof Sketch Given a challenge tuple $\langle g_1, g_2, u_1, u_2 \rangle$ we decide whether it is a random tuple or a Diffie-Hellman tuple as follows:

Step 1: Pick a random subset of k vectors $\gamma^{(1)}, \ldots, \gamma^{(k)} \subseteq \Gamma$ and create a random linear combination $\hat{\gamma} \in \mathbb{F}_q^{2k}$ of these k vectors. The vector $\hat{\gamma}$ will eventually become the private key created by the key-builder $\mathcal{P}_2$.

Step 2: Pick random vectors $\bar{a}, \bar{b} \in \mathbb{F}_q^{2k}$ such that $(\bar{a} - \bar{b}) \cdot \hat{\gamma} = 0$.

Step 3: Set $h_i = g_1^{a_i} g_2^{b_i}$ for $i = 1, \ldots, 2k$.

Step 4: Pick a random vector $\bar{\alpha} \in \mathbb{F}_q^{2k}$ such that $\bar{\alpha} \cdot (\bar{a} - \bar{b}) = 0$.

Step 5: Set $y = \prod h_i^{\alpha_i}$.

Step 6: Set $\hat{\theta} = \frac{\bar{\alpha} \cdot \bar{b}}{\hat{\gamma} \cdot \bar{b}}$.

Set $\bar{d} = \hat{\theta}\hat{\gamma} = \langle \delta_1, \ldots, \delta_{2k} \rangle$. Observe that $\bar{d}$ is a representation of y base $h_1, \ldots, h_{2k}$.

Step 7: Run the box builder $\mathcal{P}_2$ giving it the public key $\langle y, h_1, \ldots, h_{2k} \rangle$ and the representation $\bar{d}$. The box builder builds a pirate decoder $\mathcal{D}$.

Step 8: Using a random $S \in G_q$ construct the ciphertext: $\tilde{C} = \langle S,\ u_1^{a_1} u_2^{b_1},\ \ldots,\ u^{a_{2k}} v^{b_{2k}} \rangle$.

Let $\tilde{C} = \langle S, H_1, \ldots, H_{2k} \rangle$.

Step 9: Query the pirate decoder $\mathcal{D}$ on the ciphertext $\tilde{C}$. If $\mathcal{D}$ returns $S / \prod_{i=1}^{2k} H_i^{\alpha_i}$ output that $\langle g_1, g_2, u_1, u_2 \rangle$ is a Diffie-Hellman tuple. Otherwise, output $\langle g_1, g_2, u_1, u_2 \rangle$ is a random tuple.

Observe that the input $I = \langle y, h_1, \ldots, h_{2k}, \bar{d} \rangle$ given to the box-builder $\mathcal{P}_2$ is sampled from a distributed identical to the real attack. To see this note that $y, h_1, \ldots, h_{2k}$ are uniformly distributed in G_q^{2k}. Furthermore, $\bar{d}$ is a random convex combination of a random set of k private keys.

Next, note that when the challenge $\langle g_1, g_2, u_1, u_2 \rangle$ is a Diffie-Hellman tuple then given I, the ciphertext $\tilde{C}$ is a random valid ciphertext. Otherwise, given I the ciphertext $\tilde{C}$ is uniformly distributed in G_q^{2k+1}. Since the decoder behaves differently for valid and invalid ciphertexts the output in Step 9 enables us to solve the given DDH challenge. $\square$

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 [12, 14] modulo $p^2 q$ and modulo N^2. For instance, the trapdoor designed by Paillier [14] shows that if encryption is done in the group $\mathbb{F}^*_{N^2}$ then the secret factorization of N (known to the tracer only) enables the tracer to recover $\langle \delta_1, \ldots, \delta_{2k} \rangle \bmod N$ 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.

6 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 [6]. 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 $\mathbb{F}^*_p$ for some prime p where $q | p - 1$.

Key generation: Let g be a generator of G_q. Pick random $r_1, \ldots, r_{2k} \in \mathbb{F}_q$ and set $h_i = g^{r_i}$ for $i = 1, \ldots, 2k$. Next, we pick random $x_1, x_2, y_1, y_2 \in \mathbb{F}_q$ and $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{F}_q$ 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 $a \in \mathbb{F}_q$, 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 6.1. *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 $r_3, \ldots, r_{2k} \in \mathbb{F}_q$ and set $h_i = g_2^{r_i}$ for $i = 3, \ldots, 2k$. Next, choose random $x_1, x_2, y_1, y_2 \in \mathbb{F}_q$ and $\alpha_1, \ldots, \alpha_{2k} \in \mathbb{F}_q$ 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 6.1 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 [6] and is given in Appendix B. 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 6.1.

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 can then be applied for black box tracing and black box confirmation in this setting. For example, suppose a "single-key pirate" constructs a pirate decoder containing the representation $\bar{d} = (d_1, \ldots, d_{2k})$. By feeding the decoder invalid ciphertexts as above the tracer can recover $\langle g^{d_3}, \ldots, g^{d_{2k}}\rangle$. Using trapdoors of the discrete log the tracer can find $\langle d_3, \ldots, d_{2k}\rangle$ from which he can expose all active members of a coalition of size at most $k-1$. As in section 4 black box tracing is possible against any pirate by using "black box confirmation".

7 An open problem

The black box confirmation method of Section 4 gives rise to a very general black box tracing strategy. The tracing strategy works even when the pirate box has $\ell - 1$ out of the total of ℓ keys. Let $d_1, \ldots, d_\ell$ be the set of all ℓ private keys generated by the key generation algorithm. Let T_i be the set of keys $\{d_1, ..., d_i\}$. There are ℓ such sets with T_0 being the empty set and T_ℓ being the set of all private keys. Consider an encryption scheme with the following properties:

- Given a message M and an $i \in \{0, \ldots, \ell\}$ there is a set of ciphertexts CT_i such that $C \in CT_i$ decrypts correctly to M using all private keys in T_i, but decrypts to some $M' \neq M$ when using keys not in T_i (there could be a different M' for each $d \notin T_i$). Note that all of CT_ℓ are valid ciphertexts since they decrypt correctly to M under all private keys.
- An adversary given all ℓ private keys except for key d_j cannot tell whether a ciphertext C decrypts to M using key d_j.

Any encryption scheme satisfying both properties above gives rise to an efficient black box tracing algorithm. Let $CT_i(M)$ be the set of ciphertexts that decrypt to M using any key in T_i. Let CT_i be the union of $CT_i(M)$ over all M. Consider a pirate box that plays music. Define

$$p_i = \Pr_{x \in CT_i}[\text{pirate box plays music when given } x]$$

We know that $p_n = 1$ since CT_n is the set of valid ciphertexts. Similarly, we know $p_0 = 0$ since no key can decrypt messages in CT_0. Hence, since $|p_n - p_0| = 1$, there must exist a j such that $|p_j - p_{j+1}| > 1/\ell$. The only difference between CT_j and CT_{j+1} is that key d_{j+1} can decrypt ciphertexts in CT_{j+1} but cannot decrypt ciphertexts in CT_j. Hence, the pirate box is able to distinguish whether key d_j correctly decrypts the given ciphertext. By property (2) this is possible only if the pirate had d_j when building the pirate box. Hence, d_j can be pronounced guilty.

To summarize, the tracing algorithm works as follows:

Step 1: For $i = 0, ..., \ell$ estimate the value of p_i. This is done by picking many random elements in $C \in CT_i$ and testing whether the pirate box "plays music" when given C. Using $O(\ell)$ samples will produce an approximation $\hat{p}_i$ to p_i with error less than $1/4\ell$ with high probability.

Step 2: Find a j such that $|\hat{p}_j - \hat{p}_{j+1}| > 1/8\ell$. Output "$d_j$ `is guilty`" and stop.

The tracing algorithm makes a total of $O(\ell^2 \log \ell)$ probes into the pirate box. In an encryption scheme satisfying both properties above each message M can be encrypted in at least $\ell + 1$ different ways $(C_0, ..., C_\ell)$. Hence, the ciphertext length must be at least $\log_2 \ell$. Unfortunately, the current best constructions require a ciphertext whose length is linear in the number of users ℓ.

Open problem: Construct a secure encryption scheme satisfying both properties above where the ciphertext length is sub-linear in ℓ. Ideally the ciphertext length should be $O(\log \ell)$.

8 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 traitor tracing 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 traitor tracing 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.

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] D. Boneh, M. Franklin, "An efficient public key traitor tracing scheme", In proc. of Crypto '99, LNCS, Vol. 1666, Springer-Verlag, pp. 338–353, 1999.

[5] B. Chor, A. Fiat and M. Naor, "Tracing traitors", in proc. Crypto '94, pp. 257–270.

[6] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack", in proc. Crypto '98, pp. 13–25.

[7] C. Dwork, J. Lotspiech and M. Naor, "Digital signets: self-enforcing protection of digital information", in proc. of STOC '96, pp. 489–498, 1996.

[8] 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.

[9] K. Kurosawa, and Y. Desmedt, "Optimum traitor tracing and asymmetric schemes", in proc. of Eurocrypt '98, pp. 145–157.

[10] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[11] M. Naor and B. Pinkas, "Threshold traitor tracing", in proc. Crypto '98, pp. 502–517.

[12] T. Okamoto, S. Uchiyama, "A new public key cryptosystem as secure as factoring", in proc. Eurocrypt '98, pp. 308–318.

[13] 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.

[14] P. Paillier, "Public-Key Cryptosystems Based on Discrete Logarithm Residues", in proc. Eurocrypt '99, pp. 223–238.

[15] B. Pfitzmann, "Trials of traced traitors", in proc. Information Hiding Workshop, pp. 49–64, 1996.

[16] B. Pfitzmann and M. Waidner, "Asymmetric fingerprinting for larger collusions", in proc. ACM Conference on Computer and Communication Security, pp. 151–160, 1997.

[17] D. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes", *SIAM Journal on Discrete Math*, 11(1), pp. 41–53, 1998.

[18] D. Stinson and R. Wei, "Key preassigned traceability schemes for broadcast encryption", in proc. SAC '98, LNCS 1556, 1999.

A Definition of minimal access black box tracing

A minimal access black-box tracer is an oracle algorithm that takes as input a string R (representing the random bits used during key generation) and interacts with an oracle $\mathcal{P}^{\mathcal{D}}$ that implements the restricted access mechanism to the pirate decoder $\mathcal{D}$. The oracle $\mathcal{P}^{\mathcal{D}}(\cdot,\cdot)$ takes a message-ciphertext pair as input and outputs:

$$\mathcal{P}^{\mathcal{D}}(C,M) = \begin{cases} 1 & \text{if } \mathcal{D}(C) = M \\ 0 & \text{otherwise} \end{cases}$$

We write $\mathit{Trace}^{(\mathcal{P}^{\mathcal{D}})}(R)$ to denote a particular execution of the tracing algorithm. Thus, the tracing algorithm does not see the decoder's output. Instead, it can only ask whether the output is equal to some message M. The tracing algorithm outputs an index $i \in \{1, 2, \ldots, \ell\}$ or `fail`.

At the end of Section 2 we gave the intuition as to why restricting the tracer's access to the pirate decoder this way correctly models a tamper resistant pirate box. Next, we define k-resiliency of a tracing traitors system.

Black-box k-resiliency. We use the following game to define k-resiliency for a tracing traitors system. The game is between a challenger and a pirate. Both are given a security parameter s, the number of users ℓ, and a number $k < \ell$ called the collusion bound. The game runs as follows:

- The challenger runs the key generation algorithm giving it s, ℓ, and k. It obtains a public encryption key e and ℓ private keys $d_1, \ldots, d_\ell$. Next, the challenger picks a random subset $S \subset \{1, \ldots, \ell\}$ of size k. The challenger gives e and the k private keys $\{d_i \ : \ i \in S\}$ to the pirate. Let R be the random bits used by the key generation algorithm.
- The pirate outputs a pirate decoder $\mathcal{D}$ which is a probabilistic circuit that takes a ciphertext C as input and outputs a message M. For now, we assume that the pirate decoder is perfect, i.e $\Pr[\mathcal{D}(\mathit{Encrypt}(e, M)) = M] = 1$.
- The challenger now runs $\mathit{Trace}^{(\mathcal{P}^{\mathcal{D}})}(R)$ and obtains an output i where $i \in \{1, \ldots, \ell\}$ or $i = \mathtt{f}ail$.
- We say that the pirate wins the game if $i = \mathtt{f}ail$ or $i \notin S$. We denote the pirate's advantage by $\mathsf{Adv}_{\ell,k}(s)$.

Definition A.1. *We say that a tracing traitors systems is k-resilient for ℓ users if (1) for all polynomial time (in s) pirates the function $\mathsf{Adv}_{\ell,k}(s)$ is negligible and (2) the system is semantically secure as a public-key encryption scheme.*

We make three comments about this definition:

- First, notice that the challenger gives the adversary a *random* set of k secret keys. One could extend the definition to allow the adversary to adaptively choose which set of keys it gets. We do not consider such adversaries here since in the content protection scenario the adversary does not have the ability to choose which keys it gets. In fact, the distributor can assign keys to players at random so that even if the adversary chooses a specific set of players to corrupt he still only sees a random collection of secret keys.

- For simplicity we only considered pirate decoders that perfectly decrypt all valid ciphertexts. One can extend the definition to pirate decoders that only decrypt with probability ϵ. This changes nothing in the definition or in our constructions.
- The definition captures a stateless pirate decoder — by modeling the decoder as an oracle we are preventing it from keeping state between successive decryption queries. We do not consider stateful decoders in this paper.

B Proof of Theorem 6.1

To complete the proof of Theorem 6.1 we need to prove the following lemma regarding the algorithm for deciding DDH in G_q.

Lemma B.1.

a. 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.

b. 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.

Proof The proof is very similar to the one given by Cramer and Shoup [6]. Recall that the challenge ciphertext given to the adversary is $C = \langle S, H_1, \ldots, H_{2k}, v\rangle$ where

$$S = M_b \cdot u_1^{\alpha_1} u_2^{\alpha_2} \prod_{i=3}^{2k} u_2^{\alpha_i r_i} \ ; \quad H_1 = u_1 \ , \quad H_2 = u_2 \ , \quad H_3 = u_2^{r_3} \ , \quad \ldots \ , \quad H_{2k} = u_2^{r_{2k}}$$

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

We start by proving part (a). Say $\langle g_1, g_2, u_1, u_2\rangle$ is chosen from D. Then there exists an a such that $u_1 = g_1^a$ and $u_2 = g_2^a$. In this case, the challenge ciphertext C given to the adversary is a valid encryption of M_b. Indeed, one can easily verify that the challenge ciphertext C satisfies:

$$\begin{aligned} S &= y^a \\ H_i &= h_i^a \quad \text{for} \quad i = 1, \ldots, 2k \\ v &= c^a d^{\nu a} \end{aligned}$$

Furthermore, during the interaction phase with the adversary $\mathcal{A}$ the decryption oracle given to $\mathcal{A}$ behaves exactly as it does in the actual attack.

The proof of part (b) is a bit harder. Say $\langle g_1, g_2, u_1, u_2\rangle$ is chosen from R. Let $u_1 = g_1^{a_1}$ and $u_2 = g_2^{a_2}$. With high probability $a_1 \neq a_2$. Also, let $g_1 = g_2^w$ (recall $h_1 = g_1$ and $h_2 = g_2$).

We say that a ciphertext $C = \langle S, H_1, \ldots, H_{2k}, v\rangle$ is *valid* if $\log_{g_1} H_1 = \log_{g_2} H_2$. The proof of part (b) follows from the following two claims:

Claim B.2. *If the decryption oracle given to the adversary $\mathcal{A}$ rejects all invalid ciphertexts then the distribution of the bit b is independent of the adversary's view.*

Proof The value of S in the challenge ciphertext is $S = M_b \cdot u_1^{\alpha_1} u_2^{\alpha_2} \prod_{i=3}^{2k} u_2^{\alpha_i r_i}$. We show that the adversary has no information about the blinding factor B where

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

Consequently, the adversary obtains no information about M_b – the blinding factor B is a perfect one time pad.

Consider the point $Q = (\alpha_1, \ldots, \alpha_{2k})$. Given the public key the adversary knows that Q is a random point satisfying:

$$\log_{g_2} y = w\alpha_1 + \alpha_2 + \sum_{i=3}^{2k} r_i \alpha_i \pmod{q} \tag{2}$$

When the decryption oracle decrypts a ciphertext $C' = \langle S', H'_1, \ldots, H'_{2k}, v' \rangle$ it gives the adversary the value $M' = S'/(H'_1)^{\alpha_1} \cdots (H'_{2k})^{\alpha_{2k}}$. Hence, the adversary learns that the point Q satisfies the following relation:

$$\log_{g_2} \frac{S'}{M'} = \alpha_1 \log_{g_2} H'_1 + \alpha_2 \log_{g_2} H'_2 + \ldots + \alpha_{2k} \log_{g_2} H'_{2k} \tag{3}$$

Since the decryption oracle decrypts only valid ciphertexts we know that $\log_{g_1} H'_1 = \log_{g_2} H'_2$, or equivalently $\log_{g_2} H'_1 = w \log_{g_2} H'_2$. If we write $s = \log_{g_2} H'_2$ then equation (3) becomes:

$$\log_{g_2} \frac{S'}{M'} = (ws)\alpha_1 + s\alpha_2 + \alpha_3 \log_{g_2} H'_3 + \ldots + \alpha_{2k} \log_{g_2} H'_{2k} \tag{4}$$

Observe that equation (2) and all of the equations (4) obtained by the adversary are restricted to a subspace of dimension $2k-1$ (the coefficient of α_1 is always w times the coefficient of α_2).

To complete the proof of the claim note that the blinding factor B in the challenge ciphertext satisfies:

$$\log_{g_2} B = (wa_1)\alpha_1 + a_2\alpha_2 + \sum_{i=3}^{2k} (a_2 r_i)\alpha_i$$

Assuming $a_1 \neq a_2$ this relation on Q is linearly independent of equation (2) and all the equations (4). Hence, conditioned on the adversary's view, the blinding factor B is uniformly distributed in $\mathbb{F}_q$. It follows that the adversary has no information about M_b implying that its output b' is independent of b. □

Claim B.3. *The decryption oracle we supply to the adversary $\mathcal{A}$ will reject all invalid ciphertexts, except with negligible probability.*

Proof The proof is based on the fact that the adversary does not know the point $P = (x_1, x_2, y_1, y_2)$. From the public key and the challenge ciphertext the adversary knows that the point P satisfies the following relations:

$$\begin{aligned} \log_{g_2} c &= x_1 w + x_2 && (5) \\ \log_{g_2} d &= y_1 w + y_2 && (6) \\ \log_{g_2} v &= (x_1 + y_1\nu) w a_1 + (x_2 + y_2 \nu) a_2 && (7) \end{aligned}$$

Suppose the adversary queries the decryption oracle at an invalid ciphertext $C' = \langle S', H'_1, \ldots, H'_{2k}, v' \rangle$ where $H'_1 = g_1^{b_1}$ and $H'_i = g_2^{b_i}$ for $i = 2, \ldots, 2k$. Let $\nu' = \mathcal{H}(S', H'_1, \ldots, H'_{2k})$. There are three cases to consider:

Case 1: $\langle S, H_1, \ldots, H_{2k}\rangle = \langle S', H'_1, \ldots, H'_{2k}\rangle$. Since $C \neq C'$ we have $v \neq v'$. Since v' is invalid the decryption oracle will reject C'.

Case 2: $\langle S, H_1, \ldots, H_{2k}\rangle \neq \langle S', H'_1, \ldots, H'_{2k}\rangle$ and $\nu = \nu'$. This implies the adversary found a collision in the hash function $\mathcal{H}()$, contradicting the assumption on $\mathcal{H}()$.

Case 3: $\langle S, H_1, \ldots, H_{2k}\rangle \neq \langle S', H'_1, \ldots, H'_{2k}\rangle$ and $\nu \neq \nu'$. The decryption oracle will reject the ciphertext, unless

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

In other words,

$$\log_{g_2} v' = wb_1(x_1 + y_1\nu') + b_2(x_2 + y_2\nu') \tag{8}$$

However, Equations (5),(6),(7) and (8) are linearly independent. This can be seen from the determinant of

$$\det \begin{pmatrix} w & 1 & 0 & 0 \\ 0 & 0 & w & 1 \\ wa_1 & a_2 & wa_1\nu & a_2\nu \\ wb_1 & b_2 & wb_1\nu' & b_2\nu' \end{pmatrix} = w^2(\nu - \nu')(b_1 - b_2)(a_1 - a_2)$$

Since the ciphertext is invalid we know that $b_1 \neq b_2$ and hence the determinant is non zero. Equation (5), (6) and (7) tell the adversary that P lies on a certain line $\mathcal{L}$. Hence, P is one of q possible points. By linear independence, the hyperplane defined by (8) intersects the line $\mathcal{L}$ at a point. For the adversary's first query the probability (over the choice of P) that the intersection occurs at the point P is $1/q$. Hence, the invalid ciphertext will be accepted by the decryption oracle with probability at most $1/q$. After the first query is rejected the adversary knows that the point P is now one of only $q-1$ different values along $\mathcal{L}$. Hence, for the second query, the probability that $\mathcal{L}$ intersects equation (8) at P is at most $\frac{1}{q-1}$. For the i'th query the probability is at most $\frac{1}{q-i}$. Overall, the probability that an invalid ciphertext is ever accepted is at most $\sum_{i=1}^{n} \frac{1}{q-i} \leq \frac{n}{q-n}$ where n is the total number of queries. Since the adversary runs in polynomial time q is exponential in n. Hence, the probability that an invalid ciphertext is ever accepted is negligible. $\square$