Fully Scalable Public-Key Traitor Tracing

Yevgeniy Dodis Nelly Fazio
Computer Science
New York University
New York, NY, USA
{dodis,fazio}@cs.nyu.edu

Aggelos Kiayias
Computer Science
University of Connecticut
Storrs, CT, USA
aggelos@cse.uconn.edu

Moti Yung
Computer Science
Columbia University
New York, NY, USA
moti@cs.columbia.edu

ABSTRACT

Traitor Tracing Schemes constitute a very useful tool against piracy in the context of digital content broadcast. In such multi-recipient encryption schemes, each decryption key is fingerprinted and when a pirate decoder is discovered, the authorities can trace the identities of the users that contributed in its construction (called traitors). Public-key traitor tracing schemes allow for a multitude of non trusted content providers using the same set of keys, which makes the scheme "server-side scalable." To make such schemes also "client-side scalable," i.e. long lived and usable for a large population of subscribers that changes dynamically over time, it is crucial to implement efficient Add-user and Remove-user operations. Previous work on public-key traitor tracing did not address this dynamic scenario thoroughly, and there is no efficient scalable public key traitor tracing scheme that allows an increasing number of Add-user and Remove-user operations.

To address these issues, we introduce the model of Fully Scalable Public-Key Traitor Tracing, and present the first construction of such a scheme. Our model mandates for deterministic traitor tracing and an unlimited number of efficient Add-user operations and Remove-user operations. A fully scalable system achieves an unlimited number of revocations while retaining high level of efficiency by dividing the run-time of the system into periods. Each period has a saturation level for the number of revocations. When a period becomes saturated, an *efficient* new-period operation is issued by the system server that resets the saturation level. We present a formal adversarial model for our system taking into account its periodic structure, and we prove our construction secure, both against adversaries that attempt to cheat the revocation mechanism as well as against adversaries that attempt to cheat the traitor tracing mechanism.

Keywords: Digital Content Distribution, Traitor Tracing, Scalability, Broadcast Encryption, Multicast

1. INTRODUCTION

An important application of global networking is digital content distribution. For such an application (e.g., Pay-TV) to remain economically viable for the long run, it is important to design distribution schemes with certain basic properties: (1) Security– this assures a subscription-based model of exclusive content reception; (2) scalability– which assures efficient operation supporting many content providers and a dynamically changing population of subscribers; and (3) piracy protection– to prevent or deter illegal distribution.

To achieve security, a content distribution scheme requires to implement multi-user encryption mechanism that assures that only current subscribers can receive the content.

Regarding piracy protection, the state of the art method which applies to software-based platform-independent architectures, is the notion of *traitor tracing schemes* which we concentrate on in this work. A traitor tracing scheme is a multi-recipient encryption system that can be used for digital content distribution, with the property that the decryption key of each user is marked (fingerprinted). The server of the system is capable of using a traitor tracing algorithm: a procedure that given access to a pirate-decoder is capable of recovering identities of subscribers that participated in its construction (called traitors). A traitor tracing scheme is, therefore, a deterrence to piracy due to the fear of exposure.

Scalable Systems. In the context of content distribution, scalability has two facets: server-side and client-side. Server-side scalability is assured by employing a *public-key* scheme, which allows any third party to use the encryption mechanism and broadcast digital content to the set of subscribers. This is very appealing as it allows a multitude of digital-content providers (e.g. many different channels) to take advantage of the availability of secure broadcast to distribute their content without the need to maintain relationships with clients. The clients are, in fact, managed by the system server that is only responsible for maintaining and assigning the clients' decryption keys as well as publishing the encryption key. Namely, the server acts as a pure key (and account) management service. Regarding client-side scalability, observe that digital content distribution systems typically involve a large population of users (accounts), that is changing dynamically during the run-time of the system. New users should be introduced, and others need to be removed from the active user population that is capable of

receiving the digital content. To allow for a scalable management of accounts keys should be easy to generate and revoke.

To date, no schemes have been proposed that provide both client-side and server-side scalability in the context of traitor tracing schemes. This motivates us to define and realize a *Fully Scalable* Public-Key Traitor Tracing Schemes which achieves this combination.

PREVIOUS RESULTS. Traitor Tracing Schemes were introduced by Chor et al. [7], who employed a probabilistic design: each user possesses a different subset of a set of keys and tracing is achieved using the properties of the key assignment. The results of Chor et al. were later implemented with concrete combinatorial designs by [21]. These schemes do not possess a Remove-user operation. Later these results were extended by [12, 18], who also considered the combination of traitor tracing schemes with efficient revocation methods (cf. broadcast encryption, [11]). These schemes are not scalable, since (i) they do not support public-key technology in an efficient fashion, (ii) they employ combinatorial designs for the key-assignment that require a tight guess of an a-priori bound on the number of users[1], and (iii) the ciphertext size is an increasing function of the *total* number of revoked users in the system's life-time.

A "native" public-key traitor tracing scheme was introduced in [16] and [3] (the latter introduced a public-key scheme with deterministic traceability); both schemes did not consider revocation of keys. This was considered in the work of [20], however the number of revocations permitted by their public-key scheme is bounded. In particular, if the number of revocations executed in the life-time of the system exceeds the bound, this would allow previously revoked users to gain unlawful access to the system. Furthermore, the ciphertext size is linear in the revocation bound, something that prohibits (for efficiency purposes) to set the bound to a large value. Public-key traitor tracing schemes with comparable revocation capabilities as the [20]-scheme (bounded number of revocations) were also designed in [22] and [9, 10]. In all these schemes the bound on the number of revocations is proportional to the ciphertext size of the system. We remark that the scheme of [9] allows for an unlimited number of revocations, however this results in a significant degradation of the scheme's efficiency in the course of its run-time operation (with ciphertext sizes that could become proportional to the size of the user population). We note that client-side scalability was recognized as an important issue and was considered in the context of long lived broadcast encryption in [13], further, it can also be achieved in the context of multicast refresh-key [23, 6, 20]. These schemes however, do not operate in a server-scalable environment. In conclusion, to the best of our knowledge, none of the existing schemes satisfies the requirements of a Fully Scalable Public-Key Traitor Tracing Scheme.

OUR RESULTS. We introduce the first carefully formalized model and design of a fully scalable public-key traitor tracing scheme where an unlimited number of users can be added and removed efficiently from the system. Addition of users does not affect the keys of the existing users of the system, further, the design does not require an a-priori bound on the number of users. User-removal is achieved by dividing the run-time of the system into *periods*; in each period a bounded number of user removals can be executed; unlimited number of user-removals is achieved in our design by the implementation of an *efficient* change-period operation.

Our scheme allows efficient *deterministic* traitor tracing that recovers all traitors (in the non-black-box traceability setting), whereas it supports the black-box confirmation method [3], (for black-box traitor tracing model).

In a fully scalable scheme adversaries can run the Add-user protocol to introduce adversarially controlled users in the system, and they can observe the modifications to the public-key of the scheme that occur during the run-time operation of the scheme and potentially take advantage of them. We consider two types of adversaries, the ones that try to elude the traceability capability, and the ones that attempt to defeat the revocation mechanism of the system (the adversarial goal is distinct in these two cases, hence the differentiation between the two).

- Traceability Adversary: the adversary obtains some user-keys and constructs a pirate decryption device, employing the secret user-key information. We show that our construction is secure against this type of adversaries in the non-black-box traitor tracing model. Our traitor tracing algorithm is deterministic and recovers the identities of *all* traitors. Further, our scheme supports the black-box confirmation method, that allows a form of traceability in the black-box traitor tracing model, [3].
- Window Adversary: the adversary obtains some user-keys that are subsequently revoked; the adversary remains active and observes the revocation of other users of the system (in fact we allow the adversary to adaptively select which users are revoked). We show that our construction is secure against window adversaries as long as they are fully revoked in a "window" of the system's operation that has a certain length (which is specified as a system parameter).

The advantage of our fully scalable construction over previous schemes, comes from the fact that any fully revoked adversary in a window of the system's operation will, in fact, "expire." An expired adversary will be incapable of intercepting the scrambled content (in the semantic security sense) *even* if it remains active in the system (and can still choose which revocation to apply). It is the capability of our scheme to expire adversaries that allows for the enhanced functionality of an unlimited number of revocations. None of the previous public-key traitor tracing schemes with revocation capability [20, 22, 9, 10] possessed this crucial property. In Table 1 we compare our construction to previous public-key schemes.

2. OUR MODEL: SCALABLE PUBLIC-KEY TRAITOR TRACING

The run-time of a fully scalable public-key traitor tracing scheme is divided into periods. A period is an administra-

[1]Note that adding users beyond the bound would still be possible but it would be an expensive operation affecting the existing subscribers of the system.

	Ciphertext-Size	Max Traceable Collusion	Add-User	Remove-User	Adversaries Expire
[CFN94] (as PK)	$\mathcal{O}((\frac{v}{2})^3 \log n)$	$v/2$ (probabilistic-BB)	Bounded	n/a	n/a
[KD98]	v	-	Unlimited	n/a	n/a
[BF99]	v	$v/2$ (any Non-BB) + BB Confirmation	Bounded	n/a	n/a
[NP00] (PK-Scheme)	v	$v/2$ (Non-BB + specialized adversaries)	Unbounded	up to v revocations	NO
[TT01]	v	BB Confirmation	Unlimited	up to v revocations	NO
[DF02]	$\mathcal{O}(r \log n)$	unlimited	Bounded	unlimited	NO
[DF03]	v	BB Confirmation	Unlimited	up to v revocations	NO
This work	v	$v/2$ (any Non-BB) + BB Confirmation	Unlimited	up to v per period, Unlimited overall	YES

Table 1: Comparison of the main construction of this paper to previous Public-Key Traitor Tracing Schemes. Note that "BB" stands for Black-Box, and BB-Confirmation stands for the black-box confirmation method of [3] that requires exponential-time. The parameters used in the table are n=# of users, r=# of revocations. Note that "unlimited" means that any polynomial number of users (in the security parameter) can be supported.

tive unit managed based on activity and potentially time passing. A fully scalable scheme is comprised of the following basic procedures:

- Setup. An initialization procedure that is executed by the server; it outputs a public-key e.
- Broadcast Encryption. A public encryption algorithm $\mathcal{E}$ that takes as input the public-key e, and a plaintext M, and outputs a ciphertext C. The ciphertext C is distributed to a population of users through a broadcast channel.
- Decryption. A deterministic algorithm $\mathcal{D}$ that takes as input the ciphertext C, and a user's secret-key information and decrypts C.
- Add-user. It is a key-generation procedure that results in a personalized secret-key that can be used to invert the public-key e. It is executed by the server and secretly communicated to a new user of the system.
- Remove-user. A procedure $\mathcal{R}$ that given a public-key e and a user's secret-key $\vec{d}$, results in a public-key e', so that for all messages M, $\mathcal{E}(e', M)$ should be "incomprehensible" for the user holding the revoked secret-key $\vec{d}$, while non-removed users should be capable of decrypting it.

  The revocation procedure has a *saturation limit* that is an upper bound to the number of users that can be removed inside a period.
- Tracing. A procedure $\mathcal{T}$ that given the contents of a pirate-decoder outputs the identities of the traitor users whose keys are employed in the pirate-decoder.
- New-period. A procedure followed by the server that results in a special message transmitted to the active subscribers of the system that changes the period. Users removed in previous periods should be incapable of decrypting any data transmitted inside the new period. A period can be changed when the saturation limit is reached (a reactive change), or when a certain time-limit is reached (a *pro-active* change).

Scalability Objectives. The properties of the various functions of a fully scalable scheme should satisfy the following requirements:

- Efficient addition of unlimited number of users throughout the scheme's operation. Specifically, the Add-user operation should be a protocol executed between a new user and the server, that should have (i) communication independent of the size of the user-population, and (ii) it should not involve the existing users of the system in any way.
- Efficient traitor tracing of a pirate-decoder. Specifically, the tracing procedure should be polynomial-time in the number of users and the number of traitors.
- Efficient revocation of the decryption capabilities of a set of users inside a period, provided that the number of users to be removed is below the saturation-limit. Specifically, Remove-user should have time complexity independent of the number of users, and should be executed solely by the server, affecting *only* the public-key of the system.
- Efficient introduction of a new period. The communication overhead for changing a period should be independent of the number of users of the system and it should *not* require private communication channels between the server and the active users (but contrary to Remove-user it will require from users to modify their secret-keys — as a result in our model users are stateless within a period and statefull across periods).

Formal Modeling of Fully Scalable Schemes. The functionality of a fully scalable public-key traitor tracing scheme should be two-fold: on one hand, it should be capable of identifying users that participate in the construction of pirate-decoders; on the other hand, the system should be capable of revoking the decryption capabilities of "bad" users. We *formally* model the security of tracing and revocation in Section 5 and Section 6 respectively.

3. DISCRETE-LOG REPRESENTATIONS

3.1 The Intractability Assumption

Let $\mathcal{G}$ be a large cyclic multiplicative group of prime order q. Typically we assume that $\mathcal{G}$ is the subgroup of order q of $\mathbf{Z}_p^*$, where $q \mid p-1$ and p, q are large primes. We denote by $[q]$ the set $\{0, \ldots, q-1\}$. The intractability assumption we employ is the following:

DEFINITION 1. *Decisional Diffie Hellman Assumption. The Decisional Diffie Hellman (*DDH*) Assumption asserts that distinguishing in probabilistic polynomial-time the family* $R := \{\langle g, g', u, u' \rangle \mid g, g', u, u' \in \mathcal{G}\}$ *from the family* $D := \{\langle g, g', u, u' \rangle \mid g, g' \in \mathcal{G}, \log_g u = \log_{g'} u'\}$ *can only be done with negligible success probability.*

Let $h_0, h_1, \ldots, h_v$ be elements of $\mathcal{G}$ so that $h_j := g^{r_j}$ for $j = 0, \ldots, v$ with $r_0, \ldots, r_v \in [q]$. For a certain element $y := g^b$ of $\mathcal{G}$ a representation of y with respect to the base $h_0, \ldots, h_v$ is a $(v+1)$-vector $\vec{\delta} := \langle \delta_0, \ldots, \delta_v \rangle$ such that $y = h_0^{\delta_0} \ldots h_v^{\delta_v}$, or equivalently $\vec{\delta} \cdot \vec{r} = b$ where "$\cdot$" denotes the inner product of two vectors. It is well known (see [5]) that obtaining representations of a given y w.r.t. some base $h_0, \ldots, h_v$ is as hard as the discrete-log problem over $\mathcal{G}$. Furthermore, it was shown in [3] that if some adversary is given $m < v$ random representations of some y with respect to some base, then any additional representation that can be obtained has to be a "convex combination" of the given representations (a convex combination of the vectors $\vec{\delta_1}, \ldots \vec{\delta_m}$ is a vector $\sum_{\ell=1}^m \mu_\ell \vec{\delta_\ell}$ with $\sum_{\ell=1}^m \mu_\ell = 1$). However, our scheme makes use of a particular family of discrete-log representations, introduced below. In section 6 we will see how to modify the Lemma of [3] accordingly.

3.2 Leap-Vectors

We introduce a new family of discrete-log representations, called *leap-vectors*. Let P be a random polynomial of degree v over $\mathbf{Z}_q$. For some $z_1, \ldots, z_v \in \mathbf{Z}_q$ a leap-vector is a vector $\langle \alpha_0, \alpha_1, \ldots, \alpha_v \rangle$ over $\mathbf{Z}_q$ so that: $\langle \alpha_0, \alpha_1, \ldots, \alpha_v \rangle \cdot \langle 1, P(z_1), \ldots, P(z_v) \rangle = P(0)$.

DEFINITION 2. *Given* $z_1, \ldots, z_v \in \mathbf{Z}_q$, *and* $P(x) \in \mathbf{Z}_q[x]$ *the set of leap-vectors* $\mathcal{L}^P_{z_1,\ldots,z_v}$ *includes all vectors* $\vec{\alpha}$ *for which it holds that* $\vec{\alpha} \cdot \langle 1, P(z_1), \ldots, P(z_v) \rangle = P(0)$.

It follows that a leap-vector is a representation of $g^{P(0)}$ with respect to the base $g, g^{P(z_1)}, \ldots, g^{P(z_v)}$. Given any leap-vector $\vec{\alpha} := \langle \alpha_0, \ldots, \alpha_v \rangle$ for some values $z_1, \ldots, z_v$ it is possible to derive an equation on the coefficients of the polynomial $P(x) := a_0 + a_1 x + \ldots a_v x^v$ which is: $((\sum_{\ell=1}^v \alpha_\ell) - 1)a_0 + (\sum_{\ell=1}^v z_\ell \alpha_\ell) a_1 + \ldots + (\sum_{\ell=1}^v z_\ell^v \alpha_\ell) a_v = -\alpha_0$.

If one possesses a point in the graph of the polynomial P, $\langle x_i, P(x_i) \rangle$ it is possible to generate a leap-vector for the values $z_1, \ldots, z_v$ provided that $x_i \notin \{z_1, \ldots, z_v\}$ using the Lagrange interpolation coefficients $\lambda := \prod_j \frac{x_i}{x_i - z_j}$ and $\lambda_\ell := \frac{z_l}{z_\ell - x_i} \prod_{j \neq \ell} \frac{z_\ell}{z_\ell - z_j}$; now, observe that $\langle \lambda P(x_i), \lambda_1, \ldots, \lambda_v \rangle$ is a leap-vector. We will call such a vector the *leap-vector* associated to the point $\langle x_i, P(x_i) \rangle$. An important property of leap-vectors (proven in the Appendix) is the following:

PROPOSITION 3. *For a polynomial* P *and values* $z_1, \ldots, z_v$, *the knowledge of a leap-vector in* $\mathcal{L}^P_{z_1,\ldots,z_v}$ *implies knowledge of a linear equation on the coefficients of the polynomial* P *that is linearly independent from the equations on the coefficients of* P *that are defined using* $\langle z_1, P(z_1) \rangle, \ldots, \langle z_v, P(z_v) \rangle$.

As a result the possession of a leap-vector implies some knowledge about the polynomial P *beyond* what is implied by the points $\langle z_1, P(z_1) \rangle, \ldots, \langle z_v, P(z_v) \rangle$. In other words a leap-vector is the necessary information needed to *leap* from the values $P(z_1), \ldots, P(z_v)$ to the value $P(0)$.

4. OUR SCHEME

Setup. The description of a cyclic multiplicative group $\mathcal{G}$ of order q is generated. Then, two random generators $g, g' \in \mathcal{G}$ and two random polynomials $A, B \in \mathbb{F}[x]$ of degree v are selected. The parameter v has the following properties: (i) $m = v/2$ will be the *maximum traitor collusion size*, and (ii) v will be the *saturation limit*. Let $A(x) = a_0 + a_1 x + \ldots a_v x^v$ and $B(x) = b_0 + b_1 x + \ldots + b_v x^v$. The public-key of the system is set to be

$$\vec{e} = \big\langle g, g', y, \langle z_1, h_1 \rangle, \ldots, \langle z_v, h_v \rangle \big\rangle = \big\langle g, g', g^{A(0)}(g')^{B(0)}, \langle z_1, g^{A(z_1)}(g')^{B(z_1)} \rangle, \ldots, z_v, g^{A(z_v)}(g')^{B(z_v)} \rangle \big\rangle$$

where $z_1, \ldots, z_v \in_U [q]$ ("$a \in_U S$" means that a is selected uniformly at random from S). The server initiates a new period by publishing $\vec{e}$, and sets the *saturation level* L to 0. L is a system variable known to the server.

Add-user. When a new user i requests to be introduced to the system, the server transmits to the user the tuple $\langle x_i, A(x_i), B(x_i) \rangle$ (using a private channel) so that $x_i \in_U [q]$ and $x_i \notin \{z_1, \ldots, z_v\} \cup \mathcal{U}$. The set $\mathcal{U}$ is the user-registry that contains all values x_i that were selected in previous Add-user protocols. Subsequently the server records the value x_i as associated to user i and adds x_i to $\mathcal{U}$.

Encryption. The sender obtains the public-key of the system and then employs the encryption function $\mathcal{E}$ that given a plaintext $M \in \mathcal{G}$, selects a random $r \in_U [q]$ and sets the corresponding ciphertext to be: $\big\langle g^r, (g')^r, y^r \cdot M, \langle z_1, h_1^r \rangle, \ldots, \langle z_v, h_v^r \rangle \big\rangle$.

Decryption. The decryption algorithm $\mathcal{D}$ takes as input a tuple of the form $\langle x_i, A(x_i), B(x_i) \rangle$ and a ciphertext $\vec{C} = \big\langle C, C', C_0, \langle z_1, C_1 \rangle, \ldots, \langle z_v, C_v \rangle \big\rangle$. $\mathcal{D}$ first computes the leap-vectors $\vec{\alpha_i} = \langle (\alpha_i)_0, \ldots, (\alpha_i)_v \rangle$ and $\vec{\beta_i} = \langle (\beta_i)_0, \ldots, (\beta_i)_v \rangle$, that are associated to the points $\langle x_i, A(x_i) \rangle$ and $\langle x_i, B(x_i) \rangle$ with respect to the values $z_1, \ldots, z_v$. Observe that for $\ell = 1, \ldots, v$ it holds that $(\alpha_i)_\ell = (\beta_i)_\ell$. The decryption algorithm returns:

$$\frac{C_0}{C^{(\alpha_i)_0} (C')^{(\beta_i)_0} \prod_{\ell=1}^v (C_\ell)^{(\alpha_i)_\ell}}$$

If $\vec{C}$ is a properly formed ciphertext, i.e. $\vec{C} := \big\langle g^r, (g')^r, y^r \cdot M, \langle z_1, h_1^r \rangle, \ldots, \langle z_v, h_v^r \rangle \big\rangle$ then, due to the properties of the leap-vector representation, we have:

$$\mathcal{D}(\vec{C}) = \frac{g^{rA(0)} (g')^{rB(0)} \cdot M}{g^{r(\alpha_i)_0} \cdot (g')^{r(\beta_i)_0} \cdot Q} = M$$

where $Q \doteq \prod_{\ell=1}^v g^{r(\alpha_i)_\ell A(z_\ell)} \cdot \prod_{\ell=1}^v (g')^{r(\beta_i)_\ell B(z_\ell)}$.

Remove-user. Let $i_1, \ldots, i_k$ be the identities of the users to be removed, so that $L + k \leq v$. Suppose that the current public-key has the form $\vec{e} = \big\langle g, g', y, \langle z_1, h_1 \rangle, \ldots, \langle z_v, h_v \rangle \big\rangle$. The revocation procedure $\mathcal{R}$, uses the (private) user database

and modifies the current public-key $\vec{e}$ as follows:

$$\vec{e} = \big\langle g, g', y, \langle z_1, h_1\rangle, \ldots, \langle z_L, h_L\rangle, \\ \langle x_{i_1}, g^{A(x_{i_1})}(g')^{B(x_{i_1})}\rangle, \ldots, \langle x_{i_k}, g^{A(x_{i_k})}(g')^{B(x_{i_k})}\rangle, \\ \langle z_{L+k+1}, h_{L+k+1}\rangle, \ldots, \langle z_v, h_v\rangle\big\rangle.$$

Finally, the saturation level is increased to $L := L + k$.

New-period. When the saturation level L reaches the saturation limit v, the server defines a new period. First, the server broadcasts a special message "change period" (not encrypted). Note that we assume that change period is digitally signed by the server so that no third parties can maliciously initiate the new-period mode.

Let $enc : [q] \to \mathcal{G}$ be an easily invertible encoding that translates a number from $\{0, \ldots, q-1\}$ into an element of $\mathcal{G}$. If $\mathcal{G}$ is the subgroup of $\mathbf{Z}_p^*$ of oder $q = \frac{p-1}{2}$, then enc can be implemented as follows: $enc(a) := (a+1)^2 \bmod p$. It is easy to see that $enc(a) \in \mathcal{G}$ for any $a \in [q]$: this is because $\mathcal{G}$ is the subgroup of quadratic residues modulo p. The encoding function enc can be easily inverted as follows: given $b := enc(a)$ we compute the two square roots ρ_1, ρ_2 of a modulo p and define $enc^{-1}(b) = \min\{\rho_1, \rho_2\} - 1$ where min treats ρ_1, ρ_2 as integers in $[p]$.

The server selects $u \in_U [q]$ and transmits the encrypted message $C_{\mathsf{reset}} := \mathcal{E}(\vec{e}, enc(u))$ where $\vec{e}$ is the current public-key of the system. Suppose now that the current public-key has the form $\vec{e} = \big\langle g, g', y, \langle z_1, h_1\rangle, \ldots, \langle z_v, h_v\rangle\big\rangle$. The server modifies the public-key $\vec{e}$ as follows:

$$\vec{e}_{\mathsf{new}} = \big\langle g, g', g^{uA(0)}(g')^{uB(0)}, \langle z_1, g^{uA(z_1)}(g')^{uB(z_1)}\rangle, \ldots, \\ \langle z_v, g^{uA(z_v)}(g')^{uB(z_v)}\rangle\big\rangle.$$

Finally the server resets the saturation level $L := 0$, and updates the polynomials to be $A_{\mathsf{new}}(x) := u \cdot A(x)$ and $B_{\mathsf{new}}(x) := u \cdot B(x)$.

Upon receiving the signed change period message, the i-th user of the system enters a wait mode. When it receives the encrypted message $enc(u)$, the user decrypts it and decodes it using enc^{-1}. Then, modifies his secret tuple $\langle x_i, A(x_i), B(x_i)\rangle$, to $\langle x_i, u \cdot A(x_i), u \cdot B(x_i)\rangle$.

5. MODEL AND SECURITY FOR REVOCATION

Model for Revocation. The public-key traitor tracing scheme described in Section 4 withstands a more powerful type of attack than what has been considered so far in previous related work ([20, 22, 9, 10]). In such attack scenario, adversary $\mathcal{A}$ is allowed not only to join the system a bounded number of times v (equal to the *saturation level*, which is fixed as a system parameter), but also to observe and even actively affect the evolution of the system, by specifying which users should be revoked and their relative order in the sequence of revocations. We remark that this type of adversary defeats all previous public-key traitor tracing schemes with fixed ciphertext size, [20, 22, 10].

More formally, in our model the adversary interleaves, in any adaptively chosen order, two types of queries:

- Join query: it models the subscription to the system of a malicious user controlled by the adversary. To reply to such query, the server executes an Add-user operation and gives to the adversary the newly created user-key. Notice that after a Join query, the adversary obtains a valid user-key capable of recovering subsequent encrypted broadcasts.
- Revoke query: it models the revocation of a user from the system. To reply to such query, the server performs a Remove-user operation and gives $\mathcal{A}$ the new public key that results after the invalidation of the key corresponding to the user revoked.

Notice that the main constraint we impose to the adversary's behavior is that she can make at most v Join queries; no restriction is given for Revoke queries. Whenever $\mathcal{A}$ has finished collecting the amount of information she thinks she needs to maximize her chances to win the game, she outputs a pair of messages and receives back the encryption of either one with probability 1/2.

To fully appreciate the novelty of the attack scenario proposed above, recall that in previous work the only functionality conceded to $\mathcal{A}$ was to obtain the secret key of a user which was also *simultaneously* revoked from the system. In our model, such capability, usually called *corruption*, is split into two distinct operations. This clearly allows the adversary to mount more powerful attacks, and does indeed more closely model the reality, since the server not always find out about "bad" users immediately. Moreover, keeping the Join and Revoke operations distinct, allows us to impose on the adversary the (minimal) restriction of obtaining at most v user-keys, without bounding the number of Revoke queries. This constitutes a major novelty of our adversarial model, since in all previous work both the number of revoked users and the number of compromised user-keys (tied together by the definition of *corruption* query) were required to be bounded by v.

Clearly, for the challenge to the adversary not to be trivial, all the user-keys that $\mathcal{A}$ obtains through Join queries must have been rendered useless by corresponding subsequent Revoke queries. We model this necessary constraint requiring that before $\mathcal{A}$ asks for her challenge, there should have been an instant at which a Change-Period operation happened, such that during the window of v immediately preceding revocations, all the (at most v) user-keys in the adversary's possession are revoked.

Formal Model for Window Adversary. We now present a formal description of the above attack scenario. For a given value of the security parameter λ, the Setup algorithm is run and the adversary is given the public key of the system. Then $\mathcal{A}$ interleaves, in any adaptively chosen order, Join queries with Revoke queries. Each time the number of revoked users reaches the saturation limit v, a Change-Period operation is performed, updating the current public key. Let $x_{i_1}, \ldots, x_{i_s} \in \mathbb{F}_q$ be the values associated with (at most v) user-keys obtained by the adversary via Join queries. Then, before $\mathcal{A}$ queries the *Encryption Oracle*, a Change-Period operation takes place, so that the set of user-keys $\{x_{j_1}, \ldots, x_{j_v}\}$ in the window of the v revocations immediately preceding the Change-Period operation,

contains all the user-keys in the adversary's possession, i.e. $\{i_1, \ldots, i_s\} \subseteq \{j_1, \ldots, j_v\}$. When all the user-keys that $\mathcal{A}$ has obtained have been revoked, $\mathcal{A}$ is allowed to perform other Revoke queries; eventually she chooses two messages M_0 and M_1 and queries the *Encryption Oracle* to obtain her challenge. The *Encryption Oracle* selects a random bit $\sigma \in_R \{0,1\}$ and, using the current public key, broadcasts the message M_σ. Then, $\mathcal{A}$ might continue to issues Revoke queries, but eventually she needs to output a bit σ^*, which constitutes her best guess to the bit chosen by the *Encryption Oracle*.
Define $\mathcal{A}$'s advantage as $\mathsf{Adv}_{\mathcal{A}}(\lambda) = \mid \Pr(\sigma^* - \sigma) - 1/2 \mid$. We say that the public-key traitor tracing scheme is secure against window adversaries if $\mathsf{Adv}_{\mathcal{A}}(\lambda)$ is negligible.

SECURITY OF REVOCATION. We now formally prove that our public-key, fully scalable traitor tracing scheme described in Section 4 is secure against a window adversary (as defined in above). In the security proof, we will follow the same structural approach used in [10], first advocated in [8]. Starting from the actual attack scenario, we will consider a sequence of hypothetical games, all defined over the same probability space. In each game, the adversary's view is obtained in different ways, but its distribution is still indistinguishable among the games.

The security of our scheme relies on the DDH assumption as shown in the Theorem below.

THEOREM 4. *If the decisional Diffie-Hellman Problem is hard in $\mathcal{G}$, then the scheme presented above is secure against a chosen-plaintext attack.*

PROOF. *As a preliminary step, we first argue that the "window constraint" that our model imposes on the adversary's behavior suffices to invalidate all the user-keys that adversary $\mathcal{A}$ could have learned via* Join *queries. Indeed, such constraint implies that there must have been a* Change Period *operation which encrypt the "reset information" u when the current public key was of the form $\vec{e} = \langle g, g', \langle x_{j_1}, h_{j_1}\rangle, \ldots \langle x_{j_v}, h_{j_v}\rangle\rangle$ and $\{i_1, \ldots, i_s\} \subseteq \{j_1, \ldots, j_v\}$, where $x_{i_1}, \ldots, x_{i_s}$ are the values corresponding to the current user-keys that $\mathcal{A}$ holds. Hence, $\mathcal{A}$ couldn't generate the leap-vector $\vec{\alpha_i}$ and $\vec{\beta_i}$ (necessary to recover u from the broadcast) from any of the user-keys in its possession: it follows that $\mathcal{A}$ could not update any of its user-keys. Therefore, all the secret data that $\mathcal{A}$ gathered via the* Join *queries is completely useless to recover any message (and in particular $\mathcal{A}$'s challenge) encrypted after the* Change-Period *operation under consideration.*

We now define the sequence of "indistinguishable" games $\boldsymbol{G}_0, \boldsymbol{G}_1, \ldots$, where $\boldsymbol{G}_0$ is the original game, and the last game clearly gives no advantage to the adversary.

Game $\boldsymbol{G}_0$. *Recall that in $\boldsymbol{G}_0$, $\mathcal{A}$ receives the public key and adaptively asks* Join *and* Revoke *queries. Then, $\mathcal{A}$ queries the encryption oracle on (M_0, M_1) and receives back the encryption of one of them. Eventually, $\mathcal{A}$ outputs her guess $\sigma^* \in \{0,1\}$. Let T_0 be the event that $\sigma = \sigma^*$ in game $\boldsymbol{G}_0$.*

Game $\boldsymbol{G}_1$. *Game $\boldsymbol{G}_1$ is identical to game $\boldsymbol{G}_0$, except that, in $\boldsymbol{G}_1$, the encryption algorithm first picks a random $r \in_U [q]$, then defines $u \doteq g^r$ and $u' \doteq (g')^r$, and finally outputs $\langle g^r, (g')^r, y^r \cdot M, \langle z_1, u^{A(z_1)}(u')^{B(z_1)}\rangle, \ldots, \langle z_v, u^{A(z_v)}(u')^{B(z_v)}\rangle$. It is clear that such modification is just a syntactic change; hence, letting T_1 be the event that $\sigma = \sigma^*$ in game $\boldsymbol{G}_1$, it holds that* $\boxed{\Pr[T_0] = \Pr[T_1]}$.

Game $\boldsymbol{G}_2$. *To turn game $\boldsymbol{G}_1$ into game $\boldsymbol{G}_2$ we make another change to the encryption oracle used in game $\boldsymbol{G}_1$. Namely, the encryption algorithm now picks two random values $r, r' \in_U [q]$. The definition of the values u, u' changes, too: $u \doteq g^r$ and $u' \doteq (g')^{r'}$. Let T_2 be the event that $\sigma = \sigma^*$ in game $\boldsymbol{G}_2$. Notice that while in game $\boldsymbol{G}_1$ the values u and u' are obtained using the same value r, in game $\boldsymbol{G}_2$ they are totally independent, subject to $r \neq r'$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ can be used to construct a* PPT *algorithm $\mathcal{A}_1$ that distinguishes Diffie-Hellman tuples from totally random tuples with non negligible advantage. Hence,* $\boxed{\left|\Pr[T_2] - \Pr[T_1]\right| \leq \epsilon_1}$ *for some negligible ϵ_1.*

Game $\boldsymbol{G}_3$. *To define game $\boldsymbol{G}_3$, the encryption oracle is modified once again to output as challenge for the adversary the ciphertext $\langle g^r, (g')^r, y^t, \langle z_1, u^{A(z_1)}(u')^{B(z_1)}\rangle, \ldots, \langle z_v, u^{A(z_v)}(u')^{B(z_v)}\rangle$, where t has been chosen at random from $[q]$. Let T_3 be the event that $\sigma = \sigma^*$ in game $\boldsymbol{G}_3$. Because of this last change, the challenge no longer contains σ, nor does any other information in the adversary's view; therefore, we have that* $\boxed{\Pr[T_3] = \frac{1}{2}}$. *Moreover, we can prove (see Appendix, Lemma 12), that the adversary has the same chances to guess σ in both game $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$, i.e.* $\boxed{\Pr[T_3] = \Pr[T_2]}$.

Hence, combining all the intermediate results together, we can conclude that adversary $\mathcal{A}$'s advantage is negligible; more precisely: $\mathsf{Adv}_{\mathcal{A}}(\lambda) \leq \epsilon_1$. □

6. MODEL AND SECURITY FOR TRACEABILITY

The goal of a tracing algorithm is to obtain the identity of at least one of the pirates who colluded in creating a given "pirate decoder" D which, as in prior works, is assumed to be stateless. In this section we present two tracing algorithms that can be integrated within the scheme described above. The first method, a *non-black-box algorithm*, receives as input a "valid" key extracted from a pirate device, constructed using the keys of at most m users.[2] The second method, a *black-box algorithm*, repeatedly calls a *black-box confirmation* subroutine that, given a pirate decryption device and a subset of at most m suspected users, checks whether their user-keys were used to generate the key inside the device. It deterministically extracts the set of users whose user-keys were used to generate the pirate key.

6.1 Tracing Attack Scenario

Our adversary $\mathcal{A}$ operates similarly to the corresponding adversary in Section 5. Namely, after receiving the initial public key of the system, she can interleave (in any adaptively chosen order) up to m Join queries, upon which $\mathcal{A}$ receives the secret keys of the corresponding users, and a

[2]Recall, m denotes the *collusion* threshold, and should not be confused with the *revocation* threshold v discussed in Section 4; e.g., in our schemes $m \leq v/2$.

polynomial number of Revoke queries. Notice that each Revoke will change the public key, at the adversary monitors these changes as well. Also notice that the final set of "revoked" user is likely very different, and typically disjoint from the set of m "corrupted" users. At the end, $\mathcal{A}$ outputs a pirate decoder D which presumably works "well" with the current public key (denoted $PK_{\mathcal{A}}$). The job of the tracing algorithm is to find one or all of the (at most) m traitors whose keys were used to build D. The precise security guarantees depend on whether tracing is "black-box" or not. We describe both tracing methods in the next two subsections.

6.2 Non-Black-Box Algorithm

Our non-black-box tracing algorithm, presented below, is building on the results of [3, 20] and is tailored to our family of representations. Remember that a user-key is a vector $\langle \alpha, \beta, \gamma_1, \ldots, \gamma_v \rangle$ where $\langle \alpha, \gamma_1, \ldots, \gamma_v \rangle$ and $\langle \beta, \gamma_1, \ldots, \gamma_v \rangle$ are leap-vectors associated to the two points that the user receives when joins the system, and w.r.t. the values available in the public-key. Also notice that *any* such leap vector $\vec{\alpha} \in \mathcal{L}^P_{z_1,\ldots,z_v}$ (where $z_1 \ldots z_v$ are currently revoked users) will work for decrypting message encrypted with the current public key. In the non-black-box model, we make an assumption that the system can extract the secret key hidden in the illegal decoder and that this secret key must be a leap-vector $\vec{\alpha} \in \mathcal{L}^P_{z_1,\ldots,z_v}$. This assumption seems to be very reasonable, it is further justified by Proposition 3 and was previously used by [3]. It is also a-priori much less restrictive than the assumption made by [20] stating that the illegal key must be a convex linear combination of some of the traitors' keys. Luckily, Lemma 5 (whose proof is in the Appendix) shows that this seemingly more restrictive assumption actually follows shows from our initial assumption.

LEMMA 5. *If there exist a poly-time adversary $\mathcal{A}$ that, given the public key $\langle g, g', y, \langle z_1, h_1 \rangle, \ldots, \langle z_v, h_v \rangle \rangle$ and $m < v$ user-keys denoted by $\vec{\delta_1}, \ldots \vec{\delta_m}$, computes a new, valid user-key $\vec{\delta}$ that is not a convex linear combination of $\vec{\delta_1}, \ldots \vec{\delta_m}$ then the discrete-log problem over $\mathcal{G}$ is solvable.*

We now present a deterministic tracing algorithm that, using an error-correcting code subroutine, reveals the identities of the traitors that created the pirate key.

TRACING ALGORITHM. Given the pirate user-key $\vec{\delta} = \langle \alpha, \beta, \gamma_1, \ldots, \gamma_v \rangle$, denote $\vec{\delta'} \doteq \langle \gamma_1, \ldots, \gamma_v \rangle$. Let $\{z_1, \ldots, z_v\}$ be the set of all the users in the system and let $\{z_{i_1}, \ldots, z_{i_v}\}$ be the set of revoked users. Remember that the user-key of each user j is a vector of the form $\langle \lambda_j^{(j)} A(z_j) + \lambda_j^{(j)} B(z_j) + \lambda_{i_1}^{(j)}, \ldots, \lambda_{i_v}^{(j)} \rangle$, where $\lambda_j^{(j)}, \lambda_{i_1}^{(j)}, \ldots, \lambda_{i_v}^{(j)}$ are the Lagrange coefficients such that, for any polynomial $M \in \mathbf{Z}_q[x]$ of degree at most v, it holds that $\lambda_j^{(j)} M(z_j) + \lambda_{i_1}^{(j)} M(z_{i_1}) + \ldots + \lambda_{i_v}^{(j)} M(z_{i_v}) = M(0)$.

Consider the matrix A whose j^{th} row is $\langle \lambda_{i_1}^{(j)}, \ldots, \lambda_{i_v}^{(j)} \rangle$, $j = 1, \ldots, n$, (i.e. obtained projecting the user-key of the generic user j onto the last v components). By assumption, $\vec{\delta'}$ is a linear combination of at most $v/2$ user-keys. Therefore, there exist a vector $\vec{\nu}$ of Hamming weight at most $v/2$ such that $\vec{\nu} \cdot A = \vec{\delta'}$ $(\star)$. Consider the two matrices

$$B \doteq \begin{pmatrix} z_{i_1} & \ldots & z_{i_1}^v \\ \ldots & \ldots & \ldots \\ z_{i_v} & \ldots & z_{i_v}^v \end{pmatrix} \quad H \doteq \begin{pmatrix} -\lambda_1^1 z_1 & \ldots & -\lambda_1^1 z_1^v \\ \ldots & \ldots & \ldots \\ -\lambda_n^n z_n & \ldots & -\lambda_n^n z_n^v \end{pmatrix}$$

It's easy to verify that $A \cdot B = H$. Multiplying $(\star)$ by H, we get $\vec{\nu} \cdot H = \vec{\delta''}$, where $\vec{\delta''} \doteq \vec{\delta'} \cdot H$. Let $\mathcal{C}$ denote the linear code over $\mathbf{Z}_q^n$ that has H as its parity-check matrix (i.e. $\vec{c} \in \mathcal{C} \Longleftrightarrow \vec{c} \cdot H = \mathbf{0}$). Let $\lambda_1, \ldots, \lambda_n$ be the Lagrange coefficients so that $\lambda_1 M(z_1) + \ldots + \lambda_n M(z_n) = M(0)$, for all $M \in \mathbf{Z}_q[x]$ with degree$(M) < n$. In the following Lemma (whose proof is in the Appendix), we prove that $\mathcal{C}$ is a Generalized Reed-Solomon Code (GRS). For more details about Generalized Reed-Solomon Codes, see [17].

LEMMA 6. *It holds that,*

1. $\mathcal{C} = \big\{ \langle -\frac{\lambda_1}{\lambda_1^{(1)}} M(z_1), \ldots, -\frac{\lambda_n}{\lambda_n^{(n)}} M(z_n) \rangle \mid M \in \mathbf{Z}_q[x], \text{degree}(M) < n - v \big\}$.
2. *$\mathcal{C}$ is a linear code with message-rate $(n - v)/n$ and distance $v + 1$.*

Generalized Reed-Solomon Codes can be decoded efficiently by the algorithm of Berlekamp and Welch [1]. This means that for any vector $\vec{x} \in \mathbf{Z}_q^n$ for which there exists a vector $\vec{w} \in \mathcal{C}$ that disagrees with $\vec{x}$ in at most e positions with $e \leq \frac{n-(n-v)}{2} = v/2$, it holds that $\vec{w}$ is unique with this property ($\mathcal{C}$ has distance $v + 1$) and the vector $\vec{w}$ can be recovered in deterministic polynomial-time.

DESCRIPTION OF THE TRACING ALGORITHM. We describe how to reconstruct $\vec{\nu}$ given $\vec{\delta'}$ using the algebraic decoding algorithm of the linear code $\mathcal{C}$: $\vec{\nu}$ immediately reveals the indices $\{i_1, \ldots, i_m\} = \{i \mid (i \leq n) \wedge \nu_i \neq 0\}$.

First, we compute an arbitrary vector $\vec{\theta}$ that satisfies the system of equations $\vec{\theta} \cdot H = \vec{\delta'}$. Note that such $\vec{\theta}$ can be found by standard linear algebra since $\vec{\theta} \cdot H = \vec{\delta'}$ is a system of v equations with n unknowns, $n > v$, and H contains a non-singular minor of size v. It is easy to verify that the vector $\vec{w} := \vec{\theta} - \vec{\nu}$ belongs to the linear code $\mathcal{C}$: indeed, $\vec{w} \cdot H = \vec{\theta} \cdot H - \vec{\nu} \cdot H = \vec{\delta'} - \vec{\delta'} = \mathbf{0}$. As a result the vector $\vec{\theta}$ can be expressed as $\vec{\theta} = \vec{w} + \vec{\nu}$.

Provided that $m \leq v/2$, it holds that the Hamming weight of $\vec{\nu}$ is less than or equal to $v/2$ and as a result $\vec{\theta}$ is a n-vector that differs in at most $v/2$ positions from the vector $\vec{w}$ (which belongs to $\mathcal{C}$). As a result we can view $\vec{\theta}$ as a "partially corrupted" $\vec{w}$ (in at most $v/2$ positions) and employ the error correcting algorithm of the linear code $\mathcal{C}$ to recover $\vec{w}$, by running the Berlekamp-Welch decoding algorithm for GRS-codes on input $\vec{\theta}$. Then, $\vec{\nu}$ can be computed immediately as $\vec{\nu} = \vec{\theta} - \vec{w}$.

TIME-COMPLEXITY. The tracing procedure has time complexity $\mathcal{O}(n^2)$, which can be optimized further to $\mathcal{O}(n(\log n)^2)$, if matrix operations are implemented in a more sophisticated manner, see e.g. [2]. If the number of traitors exceeds the bound $v/2$ it is still possible to extract candidate sets of potential traitors, by employing GRS-decoding "beyond the

error-correction bound", the Guruswami-Sudan algorithm, [14]. This will work provided that that the size of the traitor collusion is less or equal to $n - \sqrt{n(n-v)}$.

TRACING. Suppose that the contents of a pirate-decoder are exposed. In order to decrypt, a pirate-decoder should contain a representation $\vec{\delta}$ of y w.r.t. $h_1, \ldots, h_v$. Let the identities of the traitor users be $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$, where n is the number of currently active users of the system. Due to Lemma 5, $\vec{\delta}$ should be a convex combination of the vectors $\vec{\delta_{i_1}}, \ldots, \vec{\delta_{i_m}}$ that correspond to the traitors' user-keys $\vec{\delta_{i_j}} = \langle \lambda_0^{(i_j)} A(i_j), \lambda_0^{(i_j)} B(i_j), \lambda_1^{(i_j)}, \ldots, \lambda_v^{(i_j)}$, $j = 1, \ldots, m$.

Now suppose that $\vec{\delta_1}, \ldots, \vec{\delta_n}$ are all the user-keys that were given to the subscribers of the system so far. It follows that $\vec{\delta}$ must be in the linear span of $\vec{\delta_1}, \ldots, \vec{\delta_n}$. Provided that $m \leq v/2$, we use the tracing algorithm that was presented above to obtain the identities of the traitors $\{i_1, \ldots, i_m\}$.

6.3 Black-Box Algorithm

Our Black-Box algorithm will access the decoder in a *black-box* way (namely, it cannot extract the algorithm or the keys hidden inside D, but can observe D's input-output behavior). Also similarly to previous works [4, 20, 22], our algorithm will only efficiently achieve the *black-box confirmation property*. Informally, this is a subroutine that can be used to test whether a given set Susp of at most m suspected users does include all the traitors that cooperated to construct a given pirate decoder D (and outputs at least one such pirate). On a pessimistic note, this means that our tracing algorithm might have to go through all m-element subsets of the user set to do full-fledged tracing. However, we point out that: (1) in many cases a lot of partial information about the set of corrupted users, makes the search dramatically smaller; (2) all the previous public-key traitor tracing schemes suffer from the same problem; (3) as observed in [15], the problem seems to be inherent in our setting.

However, we significantly improve the previous black-box confirmation algorithms in the following respects: (1) formal modeling of the problem; (2) our algorithm allows the adversary to *adaptively* corrupt players before building the pirate decoder; (3) our algorithm works as long as the decoder works on at least on ε fraction of correctly formed messages (rather than with probability 1; call such decoders ε-useful), where ε is the desired threshold below which the decoder is considered "useless" (following the "threshold tracing" approach of [19]); say, $\varepsilon = 0.01$.

DEFINITION 7. *A* Black-Box Confirmation *(*BBC*) algorithm is a probabilistic oracle machine, taking as oracle input a pirate decoder D, and as regular input a public key PK on which D works and a set* Susp *of suspected traitors. It outputs a user i or the special symbol $'?'$.* BBC *is said to properly work on an ε-useful decoder D output by $\mathcal{A}$, if the following two properties hold with all but negligible probability for any probabilistic polynomial time adversary $\mathcal{A}$:*

- **Confirmation:** *if* $\mathcal{B} \subseteq \mathsf{Susp}$ *then* $\mathsf{BBC}^D(PK, \mathsf{Susp})$ *outputs some $i \in \mathcal{B}$.*
- **Soundness:** *if* $\mathsf{BBC}^D(PK, \mathsf{Susp})$ *outputs $i \neq '?'$, then $i \in \mathcal{B}$.*

OUR BLACK-BOX CONFIRMATION ALGORITHM. The algorithm below assumes the collusion threshold m is at most $v/2$ (precisely, $2m - 1 \leq v$). Based on the current suspect set I (initialized to Susp) and using the secret key SK, our BBC will create an *invalid public key* $PK(I)$ (the initial value of the public key is $PK_{\mathcal{A}}$). It will then observe the behavior of D when fed encryption of the form $\mathcal{E}(PK(I), M)$ (for random M), by estimating $\delta(I) \doteq \mathsf{Succ}_{PK(I)}(D)$. We notice that the Chernoff bound implies that the latter estimation can be done quickly and accurately (by computing statistics from repeated sampling), provided $\delta(I)$ is "large enough" (specifically, at least ε/m). Now, BBC takes any index $i \in I$, and accurately estimates $\delta(I \setminus \{i\})$. If the difference between $\delta(I)$ and $\delta(I \setminus \{i\})$ is "non-trivial" (specifically, at least $\varepsilon/2m$), it proclaims i as a traitor. Otherwise, it sets $I := I \setminus \{i\}$, and repeats the entire procedure until $I = \emptyset$ (in which case it outputs $'?'$).

The last main detail to be filled in is how the algorithm generates the invalid public key $PK(I)$. Recall from Section 4 that the global secret key SK consists of two random v-degree polynomials over $\mathbb{F}_q[x]$: let (A, B) be the secret key corresponding to the initial public key $PK_{\mathcal{A}}$. Given the set I, we create two random v-degree polynomials A' and B' except they agree with A and B on points in I: for every $i \in I$, $A(i) = A'(i)$, $B(i) = B'(i)$. Notice that, since $|I| \leq m \leq v/2$, this creates no problem. We then create the public key $PK(I)$ as if the global secret key where $SK' = (A', B')$ rather than $SK = (A, B)$. Specifically, we set $y' \doteq g^{A'(0)} \cdot g'^{B'(0)}$, $h'_\ell \doteq g^{A'(z_\ell)} \cdot g'^{B'(z_\ell)}$, $\ell = 1 \ldots v$, and let $PK(I) \doteq (y', h'_1, \ldots, h'_v)$ (we omit fixed g, g', etc.).

CORRECTNESS OF TRACING. The correctness of the tracing algorithm follows from Lemma 8 and Lemma 9 below, whose proof will be given in the full version of the paper.

LEMMA 8. *Under the* DDH *assumption, if $|I| \leq m \leq z/2$ and $i \notin \mathcal{B}$, then* $|\delta(I) - \delta(I \setminus \{i\})|$ *is negligible.*

The above Lemma immediately implies the soundness of our algorithm. Namely, it can accuse an innocent user with at most a negligible probability. Informally, under the DDH assumption it is impossible to notice if the value $(A(i), B(i))$ (unknown to the adversary) was replaced by a random noise $(A'(i), B'(i))$. And the intuitive reason why we need $2m - 1 \leq v$ is that the adversary gets m shares of the secret key from users in $\mathcal{B}$, and another (at most) $m - 1$ shares are fixed by $I \setminus \{i\}$. Hence, if we had $2m - 1 > v$ (the degree of our polynomials), the adversary can do the interpolation in the exponent to check the consistency of all the values in the enabling block, easily spotting which ones are "real" and which ones are "fake". (And it turns out that the above minimal restriction is sufficient to prove Lemma 8).

LEMMA 9. *Under the* DDH *assumption, if* $\mathcal{B} \subseteq \mathsf{Susp}$ *and* $|\mathsf{Susp}| \leq m$, *then* $|\delta(\mathsf{Susp}) - \mathsf{Succ}_{PK}(D, R)|$ *is negligible.*

The above Lemma implies that if the box was useful at the start (i.e., $\mathsf{Succ}_{PK}(D) \geq \varepsilon$) and $\mathcal{B} \subseteq \mathsf{Susp}$, then the decoder cannot "notice" that PK was changed to $PK(\mathsf{Susp})$, i.e. $\delta(\mathsf{Susp}) \gtrsim \varepsilon$.[3] Coupled with the obvious fact that $\delta(\emptyset)$

[3]The relation $\gtrsim$ is meant to indicate that $\delta(\mathsf{Susp})$ is greater than ε minus negligible terms.

is negligible (since M is encrypted with a totally random one-time pad), we see that there must be a time when $\delta(I)$ changes by a non-trivial amount (i.e., at least by $\varepsilon/2m$) when we remove some $i \in I$. This i will then be output by our algorithm, and since i cannot be an innocent user (by Lemma 8), i must be one of the traitors. This shows the confirmation property.

7. MANAGEMENT OF RUN-TIME PERIOD DIVISION

The New-Period operation is invoked every v user removals (the saturation limit). Nevertheless, depending on the system operation it is of advantage to invoke the New-Period operation more frequently. In particular New-Period will expire any adversary that is contained in the window of the last v revocations (Theorem 4). Nevertheless if a New-Period is issued every v revocations it is not guaranteed that it will expire *any* adversary that is revoked in a span of any consecutive v revocations. For this reason we can employ a parameter $\alpha \in \{1, \ldots, v\}$ and issue a New-Period every α revocations. This guarantees that *any* adversary that is revoked in a span of any $v - \alpha$ consecutive revocations will expire as the following Theorem reveals (see the proof in the Appendix):

THEOREM 10. *Given $\alpha \in \{1, \ldots, v-1\}$, suppose that a* New-Period *is issued every α revocations. Then* any *window adversary that is totally revoked in a span of $v-\alpha$ revocations will expire.*

We also remark that New-Period operations can be issued after a certain timer expires. This *proactive* mode of operation can be employed in addition to the *reactive* mode of issuing a New-Period every α revocations. It increases the system's resilience to window adversaries in the following sense: assume that the timer expires after τ time units; if the expected number of revocations in τ time-units is ρ, following a similar argumentation as in Theorem 10, we can argue that our system is expected to expire *any* adversary that is totally revoked within a time span of $(\frac{v}{\rho} - 2)\tau$ time-units. A detailed probabilistic analysis will be given in the full version.

8. REFERENCES

[1] E. Berlekamp and L. Welch. Error Correction of Algebraic Block Codes, 1986. U.S. Patent, Number 4,633,470.

[2] D. Bini and V. Y. Pan. *Polynomial and Matrix Computations (vol. 1): Fundamental Algorithms*. Birkhauser-Verlag, 1994.

[3] D. Boneh and M. Franklin. An Efficient Public Key Traitor Tracing Scheme. In *Advances in Cryptology - Crypto '99*, pages 338–353. Springer-Verlag, 1999.

[4] D. Boneh and M. Franklin. An Efficient Public Key Traitor Tracing Scheme. Manuscript, full version of [3], available at `http://crypto.stanford.edu/~dabo/pubs.html`, 2001.

[5] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates — Building in Privacy*. PhD thesis, Technical University of Eindhoven, 1999.

[6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast Security: A Taxonomy and some Efficient Constructions. In *Proceedings of IEEE INFOCOM '99*, volume 2, pages 708–716, 1999.

[7] B. Chor, A. Fiat, and N. Naor. Tracing Traitors. In *Advances in Cryptology - Crypto '94*, pages 257–270. Springer-Verlag, 1994. LNCS 839.

[8] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Scheme Secure against Adaptive Chosen Ciphertext Attack. Manuscript, 2001.

[9] Y. Dodis and N. Fazio. Public Key Broadcast Encryption for Statless Receivers. In *Digital Rights Management - DRM '02*, 2002.

[10] Y. Dodis and N. Fazio. Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. In *Public Key cryptography - PKC '03*, pages 100–115. Springer-Verlag, 2003. LNCS 2567.

[11] A. Fiat and M. Naor. Broadcast Encryption. In *Advances in Cryptology - Crypto '93*, pages 480–491. Springer-Verlag, 1993. LNCS 773.

[12] E. Gafni, J. Staddon, and Y. L. Yin. Efficient Methods for Integrating Traceability and Broadcast Encryption. In *Advances in Cryptology - Crypto '99*, pages 372–387. Springer-Verlag, 1999. LNCS 1666.

[13] A Garay, J. Staddon, and A. Wool. Long-Lived Broadcast Encryption. In *Advances in Cryptology - Crypto 2000*, pages 333–352. Springer-Verlag, 2000. LNCS 1880.

[14] V. Guruswami and M. Sudan. Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes. In *IEEE Symposium on Foundations of Computer Science*, pages 28–39, 1998.

[15] A. Kiayias and M. Yung. Self Protecting Pirates and Black-Box Traitor Tracing. In *Advances in Cryptology - Crypto '01*, pages 63–79. Springer-Verlag, 2001. LNCS 2139.

[16] K. Kurosawa and Y. Desmedt. Optimum Traitor Tracing and new Direction for Asymmetricity. In *Advances in Cryptology - EuroCrypt '98*, pages 145–157. Springer-Verlag, 1998. LNCS 1403.

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

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

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

[20] M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Financial Cryptography - FC 2000*, pages 1–20. Springer-Verlag, 2000. LNCS 1962.

[21] D. R. Stinson and R. Wei. Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. *SIAM Journal on Discrete Mathematics*, 11(1):41–53, 1998.

[22] W.G. Tzeng and Z.J. Tzeng. A Public-Key Traitor Tracing Scheme with Revocation Using Dynamics Shares. In *Public Key Cryptography - PKC '01*, pages 207–224. Springer-Verlag, 2001. LNCS 1992.

[23] D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. Available at `ftp://ftp.ietf.org/rfc/rfc2627.txt`, 1997.

APPENDIX

Proof of Proposition 3

Consider the following system of equations on the coefficients $a_0, \ldots, a_v$ of the polynomial P defined by the points

$\langle z_1, P(z_1)\rangle, \ldots, \langle z_v, P(z_v)\rangle$ and the equation associated to the known α-vector:

$$\begin{pmatrix} 1 & \ldots & z_1^r \\ \vdots & \ldots & \vdots \\ 1 & \ldots & z_r^r \\ (\sum_{j=1}^r \alpha_j) - 1 & \ldots & \sum_{j=1}^r z_j^r \alpha_j \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_r \end{pmatrix} = \begin{pmatrix} P(z_1) \\ \vdots \\ P(z_r) \\ -\alpha_0 \end{pmatrix}$$

It is immediate that the equation defined by the α-vector is linearly independent to the equations defined by the points $\langle z_1, P(z_1)\rangle, \ldots, \langle z_v, P(z_v)\rangle$. □

The proofs of the following Lemma 12 is based on the same techniques used in [8]; the main tool we will use the is the following technical Lemma.

LEMMA 11. *Let k,n be integers with $1 \leq k \leq n$, and let K be a finite field. Consider a probability space with random variables $\vec{\alpha} \in K^{n\times 1}, \vec{\beta} = (\beta_1, \ldots, \beta_k)^T \in K^{k\times 1}, \vec{\gamma} \in K^{k\times 1}$, and $M \in K^{k\times n}$, such that $\vec{\alpha}$ is uniformly distributed over $K^n, \vec{\beta} = M\vec{\alpha} + \vec{\gamma}$, and for $1 \leq i \leq k$, the first i^{th} rows of M and $\vec{\gamma}$ are determined by $\beta_1, \ldots, \beta_{i-1}$. Then, conditioning on any fixed values of $\beta_1, \ldots, \beta_{k-1}$ such that the resulting matrix M has rank k, the value of β_k is uniformly distributed over K in the resulting conditional probability space.*

Let $\langle C^*, (C')^*, C_0^*, \langle z_1^*, C_1^*\rangle, \ldots \langle z_v^*, C_v^*\rangle\rangle$ be the challenge for the adversary $\mathcal{A}$ by the encryption oracle in both game $\mathbf{G}_2$ and $\mathbf{G}_3$. In what follows, we will denote with COINS the coin tosses of $\mathcal{A}$ and we define $\mathbf{X}_0 \doteq A(0) + wB(0)$ and $\mathbf{X}_\ell \doteq A(z_\ell) + wB(z_\ell)$, $\ell = 1 \ldots z$, where $w = \log_g g'$.

Proof of the Lemma stated in Theorem 4

LEMMA 12. $\Pr[T_4] = \Pr[T_3]$.

PROOF. *Consider the value $\boldsymbol{X}_0$ and the quantity $V := (\text{COINS}, w, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_z, \sigma, r^*, (r')^*)$. According to the specification of games $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$, V and $\boldsymbol{X}_0$ assume the same value in both games. Let us now consider the value $t^* = \log_g C_0^*$: unlike the previous two quantities, t^* assumes different values in the above two games. In particular, while in game $\boldsymbol{G}_2$, t^* contains information about the message M_σ, in game $\boldsymbol{G}_3$, t^* is just a random value: let us denote with $[t^*]_2$ and $[t^*]_3$ the values of t^* in game $\boldsymbol{G}_2$ and game $\boldsymbol{G}_3$, respectively.*

By definition of game $\boldsymbol{G}_2$, event T_2 solely depends on $(V, \boldsymbol{X}_0, [t^*]_2)$; similarly, by definition of game $\boldsymbol{G}_3$, event T_3 solely depends on $(V, \boldsymbol{X}_0, [t^*]_3)$. Moreover, event T_2 depends on $(V, \boldsymbol{X}_0, [t^*]_2)$ according to the same functional dependence of event T_3 upon $(V, \boldsymbol{X}_0, [t^*]_3)$. Therefore, to prove the Lemma, it suffices to show that $(V, \boldsymbol{X}_0, [t^*]_2)$ and $(V, \boldsymbol{X}_0, [t^*]_3)$ have the same distribution.

According to the specification of game $\boldsymbol{G}_3$, $[t^*]_3$ is chosen uniformly over $\mathbf{Z}_q$, independently from V and $\boldsymbol{X}_0$. Hence, to reach the thesis, it suffices to prove that the distribution of $[t^*]_2$, conditioned on V and $\boldsymbol{X}_0$, is also uniform in $\mathbf{Z}_q$. In game $\boldsymbol{G}_2$, the quantities $(V, \boldsymbol{X}_0, [t^*]_2)$ are related according to the following matrix equation:

$$\begin{pmatrix} \boldsymbol{X}_0 \\ [t^*]_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ r^* & w(r')^* \end{pmatrix}}_{T} \cdot \begin{pmatrix} A(0) \\ B(0) \end{pmatrix} + \begin{pmatrix} 0 \\ \log_g M_\sigma \end{pmatrix}$$

in which $det(T) = w(r^* - (r')^*) \neq 0$, subject to $r^* \neq (r')^*$. As soon as we fix the value of V, the matrix T is completely fixed, but the values $A(0)$ and $B(0)$ are still uniformly and independently distributed over $\mathbf{Z}_q$. Now, fixing a value for $\boldsymbol{X}_0$ also fixes a value for M_σ; hence, by Lemma 11, we can conclude that the conditioned distribution of $[t^*]_2$, w.r.t. V and $\boldsymbol{X}_0$, is also uniform over $\mathbf{Z}_q$. □

Proof of Lemma 5

Let g be a generator of $\mathcal{G}$, and let $z = g^w$. Using adversary $\mathcal{A}$, we want to show how to recover the value w. Choose two random polynomials $A(x)$ and $B(x)$ and the random values $z_1, \ldots, z_v$ (that will denote the indices of the revoked users) and $x_1, \ldots, x_m$, $m < v$ (that will denote the indices of the traitors). Set the public key to be $\langle g, z, y, \langle z_1, h_1\rangle, \ldots, \langle z_v, h_v\rangle\rangle$, where $y \doteq g^{A(0)} z^{B(0)}$ and $h_\ell \doteq g^{A(z_\ell)} z^{B(z_\ell)}, \ell = 1, \ldots, v$. The adversary is given such public key and m secret keys $\vec{\delta_1}, \ldots \vec{\delta_m}$, where $\vec{\delta_i} = \langle \lambda_0^{(i)} A(x_i), \lambda_0^{(i)} B(x_i), \lambda_1^{(i)}, \ldots, \lambda_v^{(i)}\rangle, i = 1, \ldots, m$. By assumption, $\mathcal{A}$ will eventually output $\vec{\delta} = \langle \alpha, \beta, \gamma_1, \ldots, \gamma_v\rangle$ such that $y = g^\alpha z^\beta \prod_{\ell=1}^v h_\ell^{\gamma_\ell}$. Considering discrete logs to the base g, we get the following equation:

$$A(0) + wB(0) = \alpha + \sum_{\ell=1}^v [A(z_\ell)\gamma_\ell] + w\Big(\beta + \sum_{\ell=1}^v [B(z_\ell)\gamma_\ell]\Big)$$

that can be rewritten as:

$$w\Big(\beta + \sum_{\ell=1}^v [B(z_\ell)\gamma_\ell] - B(0)\Big) = A(0) - \alpha - \sum_{\ell=1}^v [A(z_\ell)\gamma_\ell] \quad (\dagger)$$

As we argue below, the probability that the coefficient of w in (†) take the value 0 is negligible. Therefore, w.h.p. we can recover the value of w. To complete the argument, we now show that: $\Pr\left[\beta + \sum_{\ell=1}^v [B(z_\ell)\gamma_\ell] - B(0) = 0\right] = 1/q$.

Consider the matrix whose generic row is $\langle \lambda_1^{(i)}, \ldots, \lambda_v^{(i)}\rangle$, and distinguish the following two cases.

If $\langle \gamma_1, \ldots, \gamma_v\rangle$ is a linear combination of the rows of (‡) let p_i be the coefficient of the i^{th} row and consider $\beta' \doteq \lambda_0^{(1)} B(x_1) p_1 + \ldots + \lambda_0^{(m)} B(x_m) p_m$:

- if $\beta = \beta'$, then the fact that $\vec{\delta}$ is a valid user-key also implies (for symmetry) $\alpha = \alpha'$; hence, $\vec{\delta}$ is a linear combination of $\vec{\delta_1}, \ldots, \vec{\delta_m}$, that is a contradiction.
- if $\beta \neq \beta'$, let's make the following considerations. Recall that each of the m user-keys given to the adversary is in fact the mergence of the two leap-vectors $\langle \lambda_0^{(i)} A(x_i), \lambda_1^{(i)}, \ldots, \lambda_v^{(i)}\rangle$ and $\langle \lambda_0^{(i)} B(x_i), \lambda_1^{(i)}, \ldots, \lambda_v^{(i)}\rangle$. Applying the definition of leap-vector to the second leap-vector, we obtain $B(0) = \lambda_0^{(i)} B(x_i) + \lambda_1^{(i)} B(z_1) + \ldots + \lambda_v^{(i)} B(z_v)$, $i = 1, \ldots, m$. Multiplying each of these equations by the corresponding coefficient p_i and summing up memberwise, we get $\beta' + B(z_1)\gamma_1 + \ldots + B(z_v)\gamma_v$. Hence, the coefficient of w can be rewritten as $\beta - \beta' + B(0)(\sum_{i=1}^m p_i - 1)$. If the coefficient of w in the (†) were 0, then $\sum p_i \neq 1$ (since we are assuming $\beta \neq \beta'$). But this could only be the case if $B(0) = (\beta' - \beta)(\sum_{i=1}^m p_i - i)^{-1}$: since $B(0)$ is an independent random value uniformly distributed in $\mathbf{Z}_q$, this can only happen with negligible probability $1/q$.

On the other hand, if $\langle \gamma_1, \ldots, \gamma_v \rangle$ is not a linear combination of (‡) then, the coefficient of w in (†) is just a random value, and the probability that it takes the value 0 is $1/q$. □

Proof of Lemma 6
1. Let $\mathcal{C}' = \big\{ \langle -\frac{\lambda_1}{\lambda_1^{(1)}} M(z_1), \ldots, -\frac{\lambda_n}{\lambda_n^{(n)}} M(z_n) \rangle \mid M \in \mathbf{Z}_q[x], \text{degree}(M) < n - v \big\}$. If $\langle c_1, \ldots, c_n \rangle \in \mathcal{C}'$, then it is of the form $\big\langle -\frac{\lambda_1}{\lambda_1^{(1)}} M(z_1), \ldots, -\frac{\lambda_n}{\lambda_n^{(n)}} M(z_n) \big\rangle$. Then it is easy to verify that $\langle c_1, \ldots, c_n \rangle$ belongs to $\mathcal{C}$: indeed it holds that $\langle c_1, \ldots, c_n \rangle \cdot \langle -\lambda_1^{(1)} z_1^\ell, \ldots, -\lambda_n^{(n)} z_n^\ell \rangle = \sum_{i=1}^n \lambda_i M(z_i) z_i^\ell$, for any $\ell = 1, \ldots, v$. Now observe that $\sum_{i=1}^n \lambda_i M(z_i) z_i^\ell = 0$ by the choice of $\lambda_1, \ldots, \lambda_n$ (and the fact that degree$(M) < n - v$). Since $\langle -\lambda_1^{(1)} z_1^\ell, \ldots, -\lambda_n^{(n)} z_n^\ell \rangle$ is the ℓ-th column of H, it follows that $\langle c_1, \ldots, c_n \rangle \cdot H = \mathbf{0}$. This shows that $\mathcal{C}' \subseteq \mathcal{C}$. On the other hand observe that $\dim(\mathcal{C}) = n - v = \dim(\mathcal{C}')$. Since $\mathcal{C}'$ is a linear sub-space of $\mathcal{C}$ and it has the same dimension, it follows that $\mathcal{C} = \mathcal{C}'$.

2. Observe that a vector of $\mathbf{Z}_q^{n-v}$ can be encoded as the coefficients of a polynomial $M \in \mathbf{Z}_q[x]$ of degree less than $n - v$. The corresponding codeword of $\mathcal{C}$ will be the vector $\big\langle -\frac{\lambda_1}{\lambda_1^{(1)}} M(z_1), \ldots, -\frac{\lambda_n}{\lambda_n^{(n)}} M(z_n) \big\rangle$. To see that the distance of the linear code is $v + 1$ observe that any two different codewords of $\mathcal{C}$ can agree on at most $n - v - 1$ positions, or equivalently any two distinct codewords differ on at least $v + 1$ positions. □

Proof of Theorem 10 Based on Theorem 4 (semantic security against window adversary) it suffices to find a change period operation that follows a window of the system's operation (that has length v) in which all users controlled by the adversary are removed. Let $r_1, \ldots, r_{v-\alpha}$ be a sequence of revocations in which all users controlled by the adversary are removed; also let $r_{v-\alpha+1}, \ldots, r_v$ the sequence of α subsequent revocations. Since we issue a New-Period operation every α revocations, it holds that there exists exactly one $i \in \{v-\alpha+1, \ldots, v\}$ so that a new period will be introduced before revocation r_i. Now observe that for this New-Period operation, the preceding window of v revocations contains all revocations of the users controlled by the adversary, and as a result the adversary will expire. □