# Scalable Public-Key Tracing and Revoking

Yevgeniy Dodis[1], Nelly Fazio[1], Aggelos Kiayias[2], and Moti Yung[3]

[1] Computer Science Department
Courant Institute of Mathematical Sciences
New York University, New York, NY, USA
{dodis,fazio}@cs.nyu.edu

[2] Computer Science & Engineering Department
University of Connecticut, Storrs, CT, USA
aggelos@cse.uconn.edu

[3] Department of 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 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 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 the implementation of a 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 life-time of the system. New users should be introduced, and others need to be removed from the active user population entitled to receive 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 *Scalable* Public-Key Traitor Tracing Schemes which achieves this combination.

Previous Results. Traitor Tracing Schemes were introduced by Chor et al. [6], 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 [20]. These schemes do not possess a Remove-user operation. Later these results were extended by [11, 17], who also considered the combination of traitor tracing schemes with efficient revocation methods (cf. broadcast encryption, [10]). 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,[4] 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 [15, 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 [19], which described several schemes in the symmetric-key setting and a public-key scheme. In particular, one of the scheme proposed in [19] (Revocation Method 2), provides security guarantees comparable to those obtained by our scheme, but it applies to the *symmetric-key* setting, thus making it impossible for several content providers to serve the same user population without trusting each other, effectively forgoing server-side scalability (enjoyed by our scheme).

On the other hand, the *public-key* method proposed in [19] provides server-side scalability, but can only withstand a bounded number of revocations: if the number of revocations executed in the life-time of the system exceeds the bound, previously revoked users could 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 scheme in [19] (bounded number of revocations) were also designed in [21] and [8, 9]. 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 [8] allows for an unlimited number of revocations, however this results in a degradation of the scheme's efficiency in the course of its run-time operation (as ciphertext sizes also depend logarithmically on 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 [12]; it can also be achieved in the context of multicast refresh-key [22, 5, 19]. These schemes however, do not operate in a server-scalable environment. In conclusion,

[4] 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.

to the best of our knowledge, none of the existing schemes satisfies the requirements of a Scalable Public-Key Traitor Tracing Scheme.

Our Results. We introduce the first carefully formalized model of a scalable public-key traitor tracing scheme where an unlimited number of users can be added and removed efficiently from the system and we present a concrete scheme meeting these requirements, based on the DDH assumption. Addition of users does not affect the keys of the existing users of the system. Furthermore, 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* New-period operation.

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

In a 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 attempt to defeat the revocation mechanism of the system and the ones that try to elude the traceability capability. Since the adversarial goal is distinct in these two cases, we consider the following classification of adversaries:

- 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 should be 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).
- Traceability Adversary: the adversary obtains some user-keys and constructs a pirate decryption device, employing the secret user-key information (in fact we allow the adversary to adaptively select the identities of the traitors). 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. Furthermore, our scheme supports the black-box confirmation method, that allows a form of traceability in the black-box traitor tracing model, [3].

In Table 1, we compare our construction to previously proposed public-key schemes. The advantage of our scalable public-key traitor tracing scheme over previous results comes from the fact that any adversary fully revoked 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 observe and even cause other users to get revoked). 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 [19, 21, 8, 9] possessed this crucial property. Although the work of [19] described a private-key scheme providing a similar kind of functionality, achieving this in the server-scalable, public-key setting, and properly formalizing the adversarial model constitutes a technical challenge and the undertaking of this work.

## 2 Our Model: Scalable Public-Key Tracing and Revoking

The life-time of a scalable public-key traitor tracing scheme is divided into periods. A period is an administrative unit managed based on activity and potentially time passing.

A scalable scheme is comprised of the following basic procedures:

- Setup. An initialization procedure that is executed by the server, which generates a master secret key $MSK$ along with a public key $PK$; the server keeps $MSK$ secret and publishes $PK$.

|  | Ciphertext Size | Maximum Traceable Coalition Size | Add-User | Remove-User | Adversaries Expire |
|---|---|---|---|---|---|
| CFN94 [6] (as PK) | $\mathcal{O}((\frac{v}{2})^3 \log n)$ | $v/2$ (probabilistic-BB) | Bounded | N/A | N/A |
| KD98 [15] | $v$ | - | Unbounded | N/A | N/A |
| BF99 [3] | $v$ | $v/2$ (any Non-BB) + BB Confirmation | Bounded | N/A | N/A |
| NP00 [19] (PK-Scheme) | $v$ | $v/2$ (Non-BB + specialized adversaries) | Unbounded | Up to $v$ revocations | NO |
| TT01 [21] | $v$ | BB Confirmation | Unbounded | Up to $v$ revocations | NO |
| DF02 [8] | $\mathcal{O}(v \log n)$ | Unbounded | Bounded | Unbounded | NO |
| DF03 [9] | $v$ | BB Confirmation | Unbounded | Up to $v$ revocations | NO |
| This work | $v$ | $v/2$ (any Non-BB) + BB Confirmation | Unbounded | Up to $v$ per period, unbounded overall | YES |

**Table 1.** Comparison of the main construction of this paper to previous public-key traitor tracing schemes. The parameters used in the table are $n$=# of users, $v$=# of revocations. Note that "BB" stands for Black-Box, BB-Confirmation stands for the Black-Box Confirmation method of [3] that requires exponential-time, and "unbounded" means that any polynomial number of users (in the security parameter) can be supported.

- Broadcast Encryption. A public encryption algorithm $\mathcal{E}$ that takes as input the public key $PK$, and a plaintext $M$, and outputs a ciphertext $C$. The ciphertext $C$ is distributed to a population of users through an insecure broadcast channel.
- Decryption. A deterministic algorithm $\mathcal{D}$ that takes as input the ciphertext $C$, and a user's secret key and decrypts $C$.
- Add-user. It is a key-generation procedure that results in a personalized secret key $SK$ that can be used to invert the public key $PK$. It is executed by the server and secretly communicated to a new user of the system.
- Remove-user. A procedure that given a public key $PK$ and a user's secret key $SK$, results in a public key $PK'$, so that for all messages $M$, $\mathcal{E}(PK', M)$ should be "incomprehensible" for the user holding the revoked secret key $SK$, 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 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 executed by the server to initiate a fresh period, by means of transmitting (on the broadcast channel) a special message transmitted to the active subscribers of the system. Users removed in previous periods should be incapable of decrypting data subsequently transmitted within the new period. A New-period operation occurs 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 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 stateful across periods).

Formal Modeling of Scalable Schemes. The functionality of a 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 Preliminaries

Throughout the paper, $k$ will denote a security parameter; let $q$ be a $k$-bit prime number and let $\mathcal{G}$ be a large cyclic group of order $q$. We assume that $\mathcal{G}$ is the (multiplicative) subgroup of order $q$ of $\mathbb{Z}_p^*$, where $q \mid (p-1)$ and $p$ is a large prime. Alternatively, one can take as group $\mathcal{G}$ the (additive) group of points of an elliptic curve over a finite field.

**Definition 1.** *Consider the two distributions over $\mathcal{G}^4$:*

$$R \doteq \{\langle g, g', u, u' \rangle \mid g, g', u, u' \in \mathcal{G}\}$$
$$D \doteq \{\langle g, g', u, u' \rangle \mid g, g', u, u' \in \mathcal{G}, \log_g u = \log_{g'} u'\}.$$

*For all 0/1-valued probabilistic polynomial-time algorithm $\mathcal{A}$ and for all $k \in \mathbb{Z}_{\geq 0}$ , define the* DDH *advantage of $\mathcal{A}$ against $\mathcal{G}$ at $k$ as:*

$$\mathsf{AdvDDH}_{\mathcal{G},\mathcal{A}}(k) \doteq \Big| \Pr[\tau = 1 \mid \rho \xleftarrow{r} R; \tau \xleftarrow{r} \mathcal{A}(1^k, \rho)] - \Pr[\tau = 1 \mid \rho \xleftarrow{r} D; \tau \xleftarrow{r} \mathcal{A}(1^k, \rho)] \Big|.$$

*where the probability is over the random coins of $\mathcal{A}$ and the random choice of $\rho$ from $R$ and $D$, respectively.*

**Definition 2.** *Let* $\mathsf{AdvDDH}_{\mathcal{G}}(k) \doteq max_{\mathcal{A}}\mathsf{AdvDDH}_{\mathcal{G},\mathcal{A}}(k)$*, where the max is over all probabilistic, polynomial-time 0/1-valued algorithms $\mathcal{A}$.*

**Assumption 1 (Decisional Diffie-Hellman Assumption)**
*The Decisional Diffie-Hellman (*DDH*) assumption for $\mathcal{G}$ asserts that the function* $\mathsf{AdvDDH}_{\mathcal{G}}(k)$ *is negligible in $k$.*

In the following, we will also need a (weaker) assumption about the hardness of computing discrete logarithms in $\mathcal{G}$.

**Definition 3.** *For all probabilistic polynomial-time algorithm $\mathcal{A}$ and for all $k \in \mathbb{Z}_{\geq 0}$ , define the* DLog *advantage of $\mathcal{A}$ against $\mathcal{G}$ at $k$ as:*

$$\mathsf{AdvDLog}_{\mathcal{G},\mathcal{A}}(k) \doteq \Pr[w' = w \mid g, g' \xleftarrow{r} \mathcal{G}; w \leftarrow \log_g g'; w' \leftarrow \mathcal{A}(1^k, g, g')].$$

*where the probability is over the random coins of $\mathcal{A}$ and the random choice of $g, g'$ from $\mathcal{G}$.*

**Definition 4.** *Let* $\mathsf{AdvDLog}_{\mathcal{G}}(k) \doteq max_{\mathcal{A}} \mathsf{AdvDLog}_{\mathcal{G},\mathcal{A}}(k)$*, where the max is over all probabilistic, polynomial-time algorithms* $\mathcal{A}$*.*

**Assumption 2 (Discrete Logarithm Assumption)**
*The Discrete Logarithm (*DLog*) assumption for* $\mathcal{G}$ *asserts that the function* $\mathsf{AdvDLog}_{\mathcal{G}}(k)$ *is negligible in* $k$*.*

### 3.1 Discrete-Log Representations

Let $g$ be a generator of $\mathcal{G}$ and let $h_0, h_1, \ldots, h_v$ be elements of $\mathcal{G}$ such that

$$h_j = g^{r_j}$$

with $j = 0, \ldots, v$ and $r_0, \ldots, r_v \in \mathbb{Z}_q$. For a certain element $y \doteq g^b$ of $\mathcal{G}$, a representation of $y$ with respect to the base $h_0, \ldots, h_v$ is a $(v+1)$-vector

$$\boldsymbol{\delta} \doteq \langle \delta_0, \ldots, \delta_v \rangle$$

such that:

$$y = h_0^{\delta_0} \cdot \ldots \cdot h_v^{\delta_v}$$

or equivalently $\boldsymbol{\delta} \cdot \boldsymbol{r} = b$ where "$\cdot$" denotes the inner product of two vectors modulo $q$.

It is well known (e.g., see [4]) 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 Lemma 3.2 of [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 $\boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_m}$ is a vector $\sum_{\ell=1}^{m} \mu_\ell \boldsymbol{\delta_\ell}$ 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 Lemma 3.2 of [3] can be modified accordingly.

### 3.2 Leap-Vectors

We introduce a new family of discrete-log representations, called *leap-vectors*. In what follows, we denote with $\mathbb{Z}_q^v[x]$ the set of $v$-degree polynomials over $\mathbb{Z}_q$; and with $\mathbb{Z}_q^{<v}[x]$ the ring of polynomials over $\mathbb{Z}_q$ with degree less than $v$.

**Definition 5.** *Given* $z_1, \ldots, z_v \in \mathbb{Z}_q$ *and* $P(x) \in \mathbb{Z}_q^v[x]$*, the set* $\mathcal{L}^P_{z_1,\ldots,z_v}$ *of leap-vectors w.r.t.* $P(\cdot)$ *and the values* $z_1, \ldots, z_v$*, consists of all vectors* $\boldsymbol{\alpha} \in \mathbb{Z}_q^{v+1}$ *for which it holds that:*

$$P(0) = \boldsymbol{\alpha} \cdot \langle 1, P(z_1), \ldots, P(z_v) \rangle. \tag{1}$$

In other words, a leap-vector w.r.t. $P(\cdot)$ and $z_1, \ldots, z_v$, 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 $\boldsymbol{\alpha} := \langle \alpha_0, \ldots, \alpha_v \rangle$ w.r.t. some values $z_1, \ldots, z_v$, it is possible to derive the equation

$$\alpha_0 = \left(1 - \sum_{\ell=1}^{v} \alpha_\ell\right) a_0 + \sum_{j=1}^{v} \left(\sum_{\ell=1}^{v} z_\ell^j \alpha_\ell\right) a_j$$

over the coefficients of the polynomial

$$P(x) := a_0 + a_1 x + \ldots + a_v x^v.$$

If one possesses a point $\langle x_i, P(x_i) \rangle$ of the polynomial $P(\cdot)$, 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 Lagrange interpolation.

**Definition 6.** *Given distinct $x_i, z_1, \ldots, z_v \in \mathbb{Z}_q$, and $P(x) \in \mathbb{Z}_q^v[x]$, define the leap-vector $\boldsymbol{\nu}^{x_i,P}_{z_1,\ldots,z_v}$ associated to the point $\langle x_i, P(x_i)\rangle$ w.r.t. $P(\cdot)$ and $z_1, \ldots, z_v$ as:*

$$\boldsymbol{\nu}^{x_i,P}_{z_1,\ldots,z_v} \doteq \langle \lambda_0^{(i)} P(x_i), \lambda_1^{(i)}, \ldots, \lambda_v^{(i)} \rangle \tag{2}$$

*where*

$$\lambda_0^{(i)} \doteq \prod_{j=1}^{v} \frac{x_i}{x_i - z_j} \tag{3}$$

*and, for $\ell = 1, \ldots, v$*

$$\lambda_\ell^{(i)} \doteq \frac{z_\ell}{z_\ell - x_i} \cdot \prod_{\substack{j=1 \\ j \neq \ell}}^{v} \frac{z_\ell}{z_\ell - z_j}. \tag{4}$$

An important property of leap-vectors is the following:

**Proposition 1.** *Given a polynomial $P(\cdot) \in \mathbb{Z}_q^v[x]$ and the values $z_1, \ldots, z_v \in \mathbb{Z}_q$, knowledge of a leap-vector $\boldsymbol{\alpha} \in \mathcal{L}^P_{z_1,\ldots,z_v}$ implies knowledge of a linear equation on the coefficients of $P(\cdot)$ linearly independent from the linear equations defined using $\langle z_1, P(z_1)\rangle, \ldots, \langle z_v, P(z_v)\rangle$.*

*Proof.* Define

$$\boldsymbol{\pi} \doteq (P(z_1), P(z_2), \ldots, P(z_v), \alpha_0)^T.$$

The constraint on the coefficients $a_0, a_1, \ldots, a_v$ of the polynomial $P(\cdot)$ arising from points $\langle z_1, P(z_1)\rangle, \ldots, \langle z_v, P(z_v)\rangle$ and the equation associated to the leap-vector $\boldsymbol{\alpha}$, can be represented as:

$$\boldsymbol{\pi} = \mathbf{M} \cdot \boldsymbol{a}$$

where

$$\boldsymbol{a} \doteq (a_0, a_1, \ldots, a_v)^T$$

and

$$\mathbf{M} \doteq \begin{pmatrix} 1 & z_1 & \ldots & z_1^v \\ 1 & z_2 & \ldots & z_2^v \\ \vdots & \vdots & \vdots & \vdots \\ 1 & z_v & \ldots & z_v^v \\ 1 - \sum_{j=1}^{v} \alpha_j & -\sum_{j=1}^{v} \alpha_j z_j & \ldots & -\sum_{j=1}^{v} \alpha_j z_j^v \end{pmatrix}$$

Notice that matrix $\mathbf{M}$ above is obtained from a Vandermonde matrix by adding a linear combination of the first $v$ rows to the last one. Since every Vandermonde matrix has full rank, it follows that $\mathbf{M}$ has full rank, too. Hence, the equation defined by the leap-vector $\boldsymbol{\alpha}$ is linearly independent to the equations defined by the points $\langle z_1, P(z_1)\rangle, \ldots, \langle z_v, P(z_v)\rangle$. □

As a result, the possession of a leap-vector implies some knowledge about the polynomial $P(\cdot)$ *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 $A(\cdot), B(\cdot) \in \mathbb{Z}_q^v[x]$ are selected. The parameter $v$ will be also referred to as the *saturation limit*, whereas $m = \lfloor \frac{v}{2} \rfloor$ will be the *maximum traitor collusion size*. Define

$$A(x) := a_0 + a_1 x + \ldots + a_v x^v$$
$$B(x) := b_0 + b_1 x + \ldots + b_v x^v.$$

The master secret key is

$$MSK := (A(\cdot), B(\cdot))$$

and the system's public key is

$$PK := \langle g, g', g^{A(0)}g'^{B(0)}, \langle \ell, g^{A(\ell)}g'^{B(\ell)}\rangle_{\ell=1}^{v}\rangle$$

where indices $1, \ldots, v$ are used as place-holders. The server initiates a new period by publishing $PK$, 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 join the system, the server transmits (over a private channel) the tuple $\langle x_i, A(x_i), B(x_i)\rangle$ to user $i$, where

$$x_i \xleftarrow{r} \mathbb{Z}_q \quad x_i \notin \{1, \ldots, v\} \cup \mathcal{U}.$$

The set $\mathcal{U}$ is the user-registry containing all values $x_i$ that were selected in previous executions of the Add-user protocol. 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 current public key of the system

$$PK := \langle g, g', y, \langle z_1, h_1\rangle, \ldots, \langle z_v, h_v\rangle\rangle$$

(where $y = g^{A(0)}g'^{B(0)}$ and $h_\ell = g^{A(z_\ell)}g'^{B(z_\ell)}$, for some identity $z_\ell$, $\ell = 1, \ldots, v$) and then employs the encryption function $\mathcal{E}$ that, given the public key $PK$ and a plaintext $M \in \mathcal{G}$, selects a random $r \xleftarrow{r} \mathbb{Z}_q$ and sets the corresponding ciphertext to be:

$$\langle g^r, g'^r, y^r \cdot M, \langle z_1, h_1^r\rangle, \ldots, \langle z_v, h_v^r\rangle\rangle.$$

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

$$\boldsymbol{C} = \langle u, u', u'', \langle z_1, u_1\rangle, \ldots, \langle z_v, u_v\rangle\rangle.$$

$\mathcal{D}$ first computes the leap-vectors

$$\boldsymbol{\nu}_{A,i} \doteq \boldsymbol{\nu}_{z_1,\ldots,z_v}^{x_i,A} \quad \boldsymbol{\nu}_{B,i} \doteq \boldsymbol{\nu}_{z_1,\ldots,z_v}^{x_i,B}$$

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, by Definition 6 (Equations (2) and (4)), $\boldsymbol{\nu}_{A,i}$ and $\boldsymbol{\nu}_{B,i}$ agree on all components except for the first: denoting with $(\nu_{A,i})_\ell$ (respectively $(\nu_{B,i})_\ell$) the entry in $\boldsymbol{\nu}_{A,i}$ (respectively $\boldsymbol{\nu}_{B,i}$) indexed by $\ell$, it holds that $\nu_{i,\ell} \doteq (\nu_{A,i})_\ell = (\nu_{B,i})_\ell$, for $\ell = 1, \ldots, v$.

The decryption algorithm returns:

$$\mathcal{D}(\boldsymbol{C}) \doteq \frac{u''}{u^{(\nu_{A,i})_0} u'^{(\nu_{B,i})_0} \prod_{\ell=1}^{v} u_\ell^{\nu_{i,\ell}}}$$

If $\boldsymbol{C}$ is a properly formed ciphertext, i.e.

$$\boldsymbol{C} = \langle g^r, g'^r, y^r \cdot M, \langle z_1, h_1^r\rangle, \ldots, \langle z_v, h_v^r\rangle\rangle$$

then, due to the properties of the leap-vector representation (Equation (1)), we have:

$$\begin{aligned}\mathcal{D}(\boldsymbol{C}) &= \frac{g^{rA(0)}g'^{rB(0)}M}{g^{r(\nu_{A,i})_0}g'^{r(\nu_{B,i})_0}\prod_{\ell=1}^{v} g^{r\nu_{i,\ell}A(z_\ell)}g'^{r\nu_{i,\ell}B(z_\ell)}} \\ &= M\end{aligned}$$

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 is $PK = \langle g, g', y, \langle z_1, h_1\rangle, \ldots, \langle z_v, h_v\rangle\rangle$. The revocation procedure uses the user-registry $\mathcal{U}$ to retrieve the values $x_{i_1}, \ldots, x_{i_k}$ and modifies the current public key $PK$ as:

$$\begin{aligned}PK := &\langle g, g', y, \langle z_1, h_1\rangle, \ldots, \langle z_L, h_L\rangle,\\ &\langle x_{i_1}, g^{A(x_{i_1})} g'^{B(x_{i_1})}\rangle, \ldots, \langle x_{i_k}, g^{A(x_{i_k})} g'^{B(x_{i_k})}\rangle,\\ &\langle z_{L+k+1}, h_{L+k+1}\rangle, \ldots, \langle z_v, h_v\rangle\rangle.\end{aligned}$$

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

New-period. When a Remove-user operation is invoked such that the resulting saturation level $L$ would "overflow" the saturation limit $v$, the server starts a new period. First, the server broadcasts a special message change period (signed, but 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 operation.

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

The server selects $d_0, \ldots, d_v, e_0, \ldots, e_v \xleftarrow{r} \mathbb{Z}_q$ and transmits the reset message

$$\begin{aligned}C_{\mathsf{reset}} := \langle &\mathcal{E}(PK, enc(d_0)), \ldots, \mathcal{E}(PK, enc(d_v)),\\ &\mathcal{E}(PK, enc(e_0)), \ldots, \mathcal{E}(PK, enc(e_v))\rangle\end{aligned}$$

where $PK$ is the current public key of the system. Let $D(\cdot)$ be the polynomial defined by $d_0, \ldots, d_v$ and let $E(\cdot)$ be the polynomial defined by $e_0, \ldots, e_v$: namely,

$$\begin{aligned}D(x) &= d_0 + d_1 x + \ldots + d_v x^v\\ E(x) &= e_0 + e_1 x + \ldots + e_v x^v.\end{aligned}$$

At this point, the server resets the saturation level $L := 0$, updates the two secret polynomials to be:

$$\begin{aligned}A_{\mathsf{new}}(\cdot) &:= A(\cdot) + D(\cdot) \pmod q\\ B_{\mathsf{new}}(\cdot) &:= B(\cdot) + E(\cdot) \pmod q\end{aligned}$$

and modifies the public key $PK$ as follows:

$$PK_{\mathsf{new}} := \langle g, g', g^{A_{\mathsf{new}}(0)} g'^{B_{\mathsf{new}}(0)}, \langle \ell, g^{A_{\mathsf{new}}(\ell)} g'^{B_{\mathsf{new}}(\ell)}\rangle_{\ell=1}^{v}\rangle.$$

Upon receiving the signed change period message, user $i$ enters a wait-mode. When the user receives the reset message $C_{\mathsf{reset}}$, he/she decrypts all ciphertexts, decodes the coefficients $d_0, \ldots, d_v, e_0, \ldots, e_v$ using $enc^{-1}$ and forms the polynomials $D(\cdot), E(\cdot)$. Then, the user modifies his/her secret tuple $\langle x_i, A(x_i), B(x_i)\rangle$ to be the new tuple

$$\langle x_i, A(x_i) + D(x_i), B(x_i) + E(x_i)\rangle.$$

**Remark.** We notice that the efficiency of the New-period operation can be improved by using hybrid encryption. In particular, instead of computing and sending $2v + 2$ ciphertexts under the current public-key (which incurs a cost of $\mathcal{O}(v^2)$ in terms of communication), the server may pick a random session key $k$, use it to encrypt the $2v + 2$ coefficients via a secure one-time symmetric-key encryption scheme, and broadcast the resulting ciphertext together with $\mathcal{E}(PK, enc'(k))$ (where $enc'$ is a suitable encoding of session keys into elements of $\mathcal{G}$). Each non-revoked user will then be able to recover the coefficients $d_0$, …, $d_v$, $e_0$, …, $e_v$ from such reset message by first recovering the session key $k$ from the public-key ciphertext $\mathcal{E}(PK, enc'(k))$, and then using $k$ to decrypt the symmetric-key ciphertext. This will drop the communication cost to $\mathcal{O}(v)$. We omit the details.

# 5 Dealing with Revocation

## 5.1 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 [19, 21, 8, 9]. In our attack scenario, the adversary $\mathcal{A}$ is allowed not only to join the system up to 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. Notice that this type of adversary defeats all previous public-key traitor tracing schemes with fixed ciphertext size [19, 21, 9].

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 a variant of the Add-user operation, which allows the adversary to specify the identity for which she will get the decryption key, (whereas in a regular Add-user operation, the server would assign a random identity to the new user). Thus, the Join query models a more powerful adversary that can control the random choice of the server. 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 revoked user.

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 of winning the game, the corrupted users are revoked, the adversary outputs a pair of messages and receives back the encryption of either one with equal probability.

To fully appreciate the novelty of the attack scenario proposed above, recall that in the adversarial model that has been considered in previous work on public-key traitor tracing [19, 21, 8, 9], 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 does 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: previous work required both the number of revoked users and the number of compromised user-keys (tied together by the definition of *corruption* query) 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 by requiring that before asking for her challenge, $\mathcal{A}$ should enter a wait-mode during which all the (at most $v$) users she corrupted are revoked within a window of consecutive revocations that should not get interrupted by a New-period operation.

It is interesting to point here some technical similarities of the window adversary model to a (lunch-time) Chosen Ciphertext Attack (CCA1). In particular, in a lunch-time attack the adversary, prior to obtaining the challenge, can query a decryption oracle to obtain decryptions of chosen ciphertexts; in the security proof, this introduces the technical challenge of simulating such decryption oracle. In the case of a window-adversary, the adversary can query the Join oracle to obtain valid decryption keys (that will be revoked afterwards). From a technical viewpoint, simulating the Join oracle is a technical challenge of similar nature to the task of simulating the decryption oracle of a CCA1 attacker. Indeed, in our security proof and system design we take advantage of techniques that were developed for dealing with CCA1 attacks.

Formal Model for Window Adversary. We formalize the above attack scenario in terms of the window adversary attack game $\mathbf{G}^{v}_{\mathsf{win}}(1^k)$, played between a challenger and the adversary $\mathcal{A}$. This game consists of three stages, denoted respectively $\mathsf{fst}$, $\mathsf{snd}$ and $\mathsf{trd}$. To enable coordination between the three stages, at the end of each stage $\mathcal{A}$ is allowed to output a piece of state information (via the variable $aux$), which will be given as input to the next stage.

The first stage ($\mathsf{fst}$) is a learning stage, in which the adversary is allowed to obtain the secret keys of at most $v$ users and to make the system evolve via $\mathsf{Revoke}$ queries. At the end of this stage, all the corrupted users get revoked.

The second stage ($\mathsf{snd}$) is a choosing stage, in which $\mathcal{A}$ picks two messages $M_0$, $M_1$ that she deems she will be able to distinguish in the ciphertext form.

In the third stage ($\mathsf{trd}$), $\mathcal{A}$ receives a challenge ciphertext $\psi^*$, which consists of the encryption of either $M_0$ or $M_1$ with equal probability. The game ends with $\mathcal{A}$ outputting her best guess to whether $M_0$ or $M_1$ was encrypted.

1. Let $\langle PK, MSK \rangle := \mathsf{Setup}(1^k)$.
2. Let $L := 0$, $\mathsf{Corr} := \emptyset$.
3. Let $state := \langle L, PK, MSK, \mathsf{Corr} \rangle$
4. $aux \leftarrow \mathcal{A}^{\mathsf{Join}(state,\cdot),\mathsf{Revoke}(state,\cdot)}(\mathsf{fst}, state.PK)$.
5. If $L + |\mathsf{Corr}| > v$ then $\mathsf{exit}$.
6. For all $x_j \in \mathsf{Corr}$ do $aux := aux || \mathsf{Revoke}(state, x_j)$.
7. $\langle aux, M_0, M_1 \rangle \leftarrow \mathcal{A}^{\mathsf{Revoke}(state,\cdot)}(\mathsf{snd}, aux, state.PK)$.
8. $\psi^* \leftarrow \mathcal{E}(state.PK, M_{\sigma^*})$, where $\sigma^* \xleftarrow{r} \{0,1\}$.
9. $\sigma \leftarrow \mathcal{A}^{\mathsf{Revoke}(state,\cdot)}(\mathsf{trd}, aux, state.PK, \psi^*)$.
10. Output $\mathsf{Success}$ if and only if $\sigma = \sigma^*$.

The two oracles employed above are defined as follows:

$\mathsf{Join}(state, x)$ :
(i) parse $state$ as $\langle L, PK, MSK, \mathsf{Corr} \rangle$;
(ii) parse $PK$ as $\langle g, g', y, \langle z_1, h_1 \rangle, \ldots, \langle z_v, h_v \rangle \rangle$;
(iii) parse $MSK$ as $(A(\cdot), B(\cdot))$;
(iv) if $x \in \{1, \ldots, v\}$, then $\mathsf{exit}$;
(v) set $\mathsf{Corr} := \mathsf{Corr} \cup \{x\}$ and return $(A(x), B(x))$.

$\mathsf{Revoke}(state, x)$ :
(i) parse $state$ as $\langle L, PK, MSK, \mathsf{Corr} \rangle$;
(ii) parse $PK$ as $\langle g, g', y, \langle z_1, h_1 \rangle, \ldots, \langle z_v, h_v \rangle \rangle$;
(iii) parse $MSK$ as $(A(\cdot), B(\cdot))$;
(iv) if $x \in \mathsf{Corr}$, then $\mathsf{exit}$;
(v) if $L = v$ then a $\mathsf{New\text{-}period}$ operation is executed and $state$ is updated accordingly (i.e., $L$ is reset to 0, $state.MSK$ is modified by adding the randomizing polynomials and $state.PK$ changes correspondingly);
(vi) set $L := L + 1$;
(vii) update $state.PK$ by replacing the pair $\langle z_L, h_L \rangle$ with $\langle x, g^{A(x)} g'^{B(x)} \rangle$;
(viii) output $state.PK$; if step (v) caused a $\mathsf{New\text{-}period}$ operation, then also output the corresponding $\mathsf{reset}$ message $C_{\mathsf{reset}}$.

Note that w.l.o.g. we assume that the adversary never corrupts the same user twice, as there is no extra information to be gained, and never revokes the users it corrupts, as they get explicitly revoked at step 6. of the attack game.

**Definition 7.** *Define $\mathcal{A}$'s advantage as*

$$\mathsf{Adv}_{\mathcal{A}}(k) \doteq \mid \Pr(\sigma = \sigma^*) - 1/2 \mid .$$

*A public-key traitor tracing scheme is secure against window adversaries if for any* $\mathsf{PPT}$ *adversary $\mathcal{A}$,* $\mathsf{Adv}_{\mathcal{A}}(k)$ *is negligible in $k$ .*

### 5.2 Security of Revocation

We now formally prove that the scalable public-key traitor tracing scheme described in Section 4 is secure against window adversaries (as defined above). In the security proof, we will follow the same structural approach used in [9], first advocated in [7]. 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 $\mathsf{DDH}$ assumption (Assumption 1) as shown below in Theorem 1.

**Theorem 1.** *Under the decisional Diffie-Hellman Assumption for $\mathcal{G}$, the scheme presented above is secure against window adversaries.*

*Proof.* We define a sequence of "indistinguishable" games $\mathbf{G}_0, \mathbf{G}_1, \ldots$, all operating over the same underlying probability space. Starting from the actual adversarial game $\mathbf{G}_0 = \mathbf{G}^{v}_{\mathsf{win}}(1^k)$, we incrementally make slight modifications to the behavior of the oracles, thus changing the way the adversary's view is computed, while maintaining the views' distributions indistinguishable among the games. In the last game, it will be clear that the adversary has (at most) a negligible advantage; by the indistinguishability of any two consecutive games, it will follow that also in the original game the adversary's advantage is negligible. Recall that in each game $\mathbf{G}_j$, the goal of adversary $\mathcal{A}$ is to output $\sigma \in \{0,1\}$ which is her best guess to the bit $\sigma^*$ used at step 7. of the attack game $\mathbf{G}^{v}_{\mathsf{win}}(1^k)$ to create the challenge ciphertext $\psi^*$: let $T_j$ be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_j$ (i.e., the event that the game ends with $\mathsf{Success}$ as output). W.l.o.g., in the following we assume that the adversary corrupts exactly $v$ users during the attack game.

**Game $\mathbf{G}_0$.** Define $\mathbf{G}_0$ to be the original game $\mathbf{G}^{v}_{\mathsf{win}}(1^k)$.

**Game $\mathbf{G}_1$.** Define the "special" $\mathsf{New\text{-}period}$ operation to be the first one to be caused by the $\mathsf{Revoke}$ oracle at step 7. of the attack game. Depending on the adversary's strategy, such "special" $\mathsf{New\text{-}period}$ operation may not occur at all.

Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that, in $\mathbf{G}_1$, the $\mathsf{reset}$ message output by the "special" $\mathsf{New\text{-}period}$ operation contains $2v + 2$ encryptions of random elements of $\mathbb{Z}_q$, rather than encryptions of the coefficients of the randomizing polynomials. This modification suggests that the secret polynomials which are contained in $state.MSK$ at the beginning of the period initiated by the "special" $\mathsf{New\text{-}period}$ operation are totally random, even given all the information in the adversary's view.

In Lemma 2 (whose proof is given below), we show that the chances of adversary $\mathcal{A}$ winning game $\mathbf{G}_1$ cannot be significantly better than her chances of winning game $\mathbf{G}_0$: more precisely,

$$\big|\Pr[T_1] - \Pr[T_0]\big| \leq (4v+4)\ \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{5}$$

**Game $\mathbf{G}_2$.** To turn game $\mathbf{G}_1$ into game $\mathbf{G}_2$, step 8. of the attack game is modified as follows:

$$8'.\ \psi^* \leftarrow \mathcal{E}(state.PK, M), \text{where } M \overset{r}{\leftarrow} \mathcal{G}, \sigma^* \overset{r}{\leftarrow} \{0,1\}$$

Because of this change, the challenge ciphertext $\psi^*$ no longer contains $\sigma^*$, nor does any other information in the adversary's view; therefore,

$$\Pr[T_2] = \frac{1}{2}. \tag{6}$$

In Lemma 3, proven below, we show that the adversary has almost the same chances to guess $\sigma^*$ in game $\mathbf{G}_1$ and $\mathbf{G}_2$: more precisely,

$$\big|\Pr[T_2] - \Pr[T_1]\big| \leq 2\ \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{7}$$

Combining Equations (5), (6), and (7) together, adversary $\mathcal{A}$'s advantage can be bounded as:

$$\mathsf{Adv}_{\mathcal{A}}(k) \leq (4v+6)\ \mathsf{AdvDDH}_{\mathcal{G}}(k).$$

□

The core of the proof of Theorem 1 is in the two lemmas that follow, Lemma 2 and Lemma 3.

Overview of the Proof Technique. Throughout the paper, we make extensive use of a technical lemma, stated and proved as Lemma 9 in [7]. For ease of reference, we report it verbatim below.

**Lemma 1.** *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 $\boldsymbol{\alpha} \in K^{n\times 1}, \boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)^T \in K^{k\times 1}, \boldsymbol{\gamma} \in K^{k\times 1}$, and $\boldsymbol{M} \in K^{k\times n}$, such that $\boldsymbol{\alpha}$ is uniformly distributed over $K^n, \boldsymbol{\beta} = \boldsymbol{M}\boldsymbol{\alpha} + \boldsymbol{\gamma}$, and for $1 \leq i \leq k$, the first ith rows of $\boldsymbol{M}$ and $\boldsymbol{\gamma}$ 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 $\boldsymbol{M}$ has rank $k$, the value of $\beta_k$ is uniformly distributed over $K$ in the resulting conditional probability space.*

Our use of this technical lemma is quite uniform across the proofs to follow. In all cases, our main aim will be to prove that some quantity $\mathsf{rand} \in \mathbb{Z}_q$ looks uniformly random to the adversary, despite all the other information in the adversary's view. At a high level, our approach is organized in the following steps.

First, we consider all the randomness underlying a specific execution of the attack game. This will include, for instance, the random coins of the adversary, the randomness used in creating the challenge, etc. We then partition all the randomness in two parts: a quantity $\boldsymbol{V}$ and a vector $\boldsymbol{\alpha}$, such that conditioning on any fixed value of $\boldsymbol{V}$, $\boldsymbol{\alpha}$ is still distributed uniformly at random in the appropriate vector space (which usually will have $\mathbb{Z}_q$ as support).

Second, we consider another vector $\boldsymbol{\beta}$, whose last entry is $\mathsf{rand}$, with the property that fixing a value for $\boldsymbol{V}$ and $\boldsymbol{\beta}$ also fixes the value of $\boldsymbol{\alpha}$, and thus all the information of the entire game (which in particular includes the information in the adversary's view).

Third, we define a matrix $\mathbf{M}$ (and possibly a vector $\boldsymbol{\gamma}$) describing the constraints binding vector $\boldsymbol{\alpha}$ to vector $\boldsymbol{\beta}$, thus obtaining a matrix equation of the form:

$$\boldsymbol{\beta} = \mathbf{M} \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}.$$

Finally, we make sure that the preconditions of Lemma 1 are fulfilled; it will follow that the last entry of $\boldsymbol{\beta}$ (which is the quantity of interest $\mathsf{rand}$), is distributed uniformly at random in $\mathbb{Z}_q$, even conditioning on fixed values of $\boldsymbol{V}$ and of all the other entries of $\boldsymbol{\beta}$, or equivalently, conditioning on all the other information in the adversary's view.

Notation. In what follows, we refer to the period initiated by the $t$th $\mathsf{New\text{-}period}$ operation as the $t$th period. Also, for notational convenience, we denote with $D^t(\cdot)$ and $E^t(\cdot)$ the randomizing polynomials chosen during the $t$th $\mathsf{New\text{-}period}$ operation and with $d_0^t, \ldots, d_v^t$ and $e_0^t, \ldots, e_v^t$ the corresponding coefficients. In some cases, it will be convenient to denote these $2v + 2$ coefficients with a uniform notation; for this reason, for $j = 1, \ldots, 2v + 2$, we additionally define $c_j^t$ as follows:

$$c_j^t \doteq \begin{cases} d_{j-1}^t & \text{if } j \in \{1, \ldots, v+1\} \\ e_{j-v-2} & \text{if } j \in \{v+2, \ldots, 2v+2\} \end{cases}$$

Moreover, let $A^t(\cdot)$ and $B^t(\cdot)$ be the values of the secret polynomials after the changes due to the $t$th $\mathsf{New\text{-}period}$ operation. In other words, the system starts with period number 0, $A^0(\cdot)$ and $B^0(\cdot)$ are the polynomials initially output by the $\mathsf{Setup}$ algorithm and

$$A^t(\cdot) \doteq A^{t-1}(\cdot) + D^t(\cdot) \quad B^t(\cdot) \doteq B^{t-1}(\cdot) + E^t(\cdot). \tag{8}$$

Also define

$$D^{t_1,t_2}(\cdot) \doteq \sum_{t=t_1}^{t_2} D^t(\cdot) \quad E^{t_1,t_2}(\cdot) \doteq \sum_{t=t_1}^{t_2} E^t(\cdot). \tag{9}$$

**Lemma 2.** $\big|\Pr[T_1] - \Pr[T_0]\big| \leq (4v + 4)\ \mathsf{AdvDDH}_{\mathcal{G}}(k)$.

*Proof.* Recall that $\mathbf{G}_1$ differs from $\mathbf{G}_0$ only in the way the $\mathsf{reset}$ message is computed for the "special" $\mathsf{New\text{-}period}$ operation: hence, if the adversary's strategy does not cause any $\mathsf{New\text{-}period}$ operation to occur during step 7. of the attack game, the two games are identical, so that in fact $\Pr[T_1] = \Pr[T_0]$, and the Lemma immediately follows.

We now discuss the case in which the "special" $\mathsf{New\text{-}period}$ operation takes place: in particular, let $\hat{t}$ be the period initiated by this operation and $D^{\hat{t}}(\cdot)$ and $E^{\hat{t}}(\cdot)$ be the randomizing polynomials used in such $\mathsf{New\text{-}period}$ operation. We then consider the sequence of $2v+3$ hybrid games $\mathbf{G}_{0,0}, \ldots, \mathbf{G}_{0,2v+2}$, where $\mathbf{G}_{0,i}$ is defined as $\mathbf{G}_0$, except that the first $i$ ciphertexts in the "special" $\mathsf{reset}$ message contain random values rather than coefficients of the randomizing polynomials $D^{\hat{t}}(\cdot)$ and $E^{\hat{t}}(\cdot)$. In other words, $\mathbf{G}_{0,0} \equiv \mathbf{G}_0$, $\mathbf{G}_{0,2v+2} \equiv \mathbf{G}_1$ and two consecutive hybrid games $\mathbf{G}_{0,i}$ and $\mathbf{G}_{0,i+1}$ differ only in that the $(i+1)$th ciphertext of the "special" $\mathsf{reset}$ message contains the $(i+1)$th coefficient in game $\mathbf{G}_{0,i}$, whereas it contains a random value in game $\mathbf{G}_{0,i+1}$. Then, to prove the Lemma it suffices to show that for all $i = 0, \ldots, 2v+1$ it holds:

$$\left|\Pr[T_{0,i+1}] - \Pr[T_{0,i}]\right| \leq 2\ \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{10}$$

To this aim, fix $i$ and consider the additional games $\mathbf{G}^0_{0,i} \equiv \mathbf{G}_{0,i}$, $\mathbf{G}^1_{0,i}$, $\mathbf{G}^2_{0,i}$, $\mathbf{G}^3_{0,i}$, $\mathbf{G}^4_{0,i} \equiv \mathbf{G}_{0,i+1}$, defined as follows:

**Game $\mathbf{G}^1_{0,i}$.** It operates as $\mathbf{G}^0_{0,i}$, except that the $(i+1)$th ciphertext in the "special" $\mathsf{reset}$ message is computed as:

$$\langle u, u', u'', \langle z_\ell, u^{A^{\hat{t}-1}(z_\ell)} u'^{B^{\hat{t}-1}(z_\ell)}\rangle_{\ell=1}^{v}\rangle$$

where $u \doteq g^r$, $u' \doteq g'^r$, $u'' \doteq u^{A^{\hat{t}-1}(0)}\ u'^{B^{\hat{t}-1}(0)} enc(c^{\hat{t}}_{i+1})$, $r \xleftarrow{r} \mathbb{Z}_q$ and $c^{\hat{t}}_{i+1}$ is either the $(i+1)$th coefficient of the randomizing polynomial $D^{\hat{t}}(\cdot)$ (if $0 \leq i \leq v$) or the $(i-v)$th coefficient of $E^{\hat{t}}(\cdot)$ (if $v+1 \leq i \leq 2v+1$). Since such modification is just a syntactic change, it holds:

$$\Pr[T^1_{0,i}] = \Pr[T^0_{0,i}]. \tag{11}$$

**Game $\mathbf{G}^2_{0,i}$.** To turn game $\mathbf{G}^1_{0,i}$ into game $\mathbf{G}^2_{0,i}$ we make another change to the way in which the $(i+1)$th ciphertext in the "special" $\mathsf{reset}$ message is computed. Namely, the value $u'$ is now computed as $u' \doteq g'^{r'}$, for a random $r' \in \mathbb{Z}_q$ such that $r' \neq r$. In other words, in game $\mathbf{G}^2_{0,i}$ the values $u$ and $u'$ are nearly independent (being subject only to $r \neq r'$), whereas in game $\mathbf{G}^1_{0,i}$ they are obtained using the same value $r$. Therefore, using a standard reduction argument, any difference in behavior between games $\mathbf{G}^1_{0,i}$ and $\mathbf{G}^2_{0,i}$ can be used to distinguish Diffie-Hellman tuples from totally random tuples. Hence,

$$\left|\Pr[T^2_{0,i}] - \Pr[T^1_{0,i}]\right| \leq\ \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{12}$$

Note that for simplicity here (and throughout the rest of the paper) we omit the negligible additive term that is caused by the negligibly-rare event $r = r'$.

**Game $\mathbf{G}^3_{0,i}$.** To define game $\mathbf{G}^3_{0,i}$, we again modify the $(i+1)$th ciphertext in the "special" $\mathsf{reset}$ message: specifically, the value $u''$ is now computed as $g^{r''}$, for a random $r'' \in \mathbb{Z}_q$.

We want to show that this modification does not alter the behavior of adversary $\mathcal{A}$ or, more precisely, that $\Pr[T^3_{0,i}] = \Pr[T^2_{0,i}]$. To this aim, we first consider all the random variables affecting the adversary's view, and then we show that they are distributed according to the same joint distribution in both games.

Let $\bar{t}$ be the total number of $\mathsf{New\text{-}period}$ operations that occur during the entire game, and for $t = 1, \ldots, \bar{t}$, let $c^t_1, \ldots, c^t_{2v+2}$ be the coefficients of the randomizing polynomials $D^t(\cdot)$ and $E^t(\cdot)$ used in the $t$th $\mathsf{New\text{-}period}$ operation. For $t = 1, \ldots, \bar{t}$, $t \neq \hat{t}$, and $j = 1, \ldots, 2v+2$, let $r^t_j$ be the randomness used to encrypt (the encoding of) coefficient $c^t_j$ in the $t$th $\mathsf{reset}$ message.

As for the "special" $\mathsf{reset}$ message (i.e., the one corresponding to $t = \hat{t}$), recall that in both game $\mathbf{G}^2_{0,i}$ and game $\mathbf{G}^3_{0,i}$, the first $i$ ciphertexts consists of just random values $s_1, \ldots, s_i \in \mathcal{G}$, rather than (the encoding of) the corresponding coefficients $c^{\hat{t}}_1, \ldots, c^{\hat{t}}_i$. Coefficients $c^{\hat{t}}_{i+2}, \ldots, c^{\hat{t}}_{2v+2}$, instead, are regularly encrypted under the public key $PK^{\hat{t}-1}$ in both games: let $r_j^{\hat{t}}$ be the randomness used in such encryptions, for $j = i+2, \ldots, 2v+2$. The ciphertext corresponding to coefficient $c_{i+1}$ in the "special" reset message constitutes the only difference between the adversary's view in game $\mathbf{G}_{0,i}^2$ and $\mathbf{G}_{0,i}^3$. In particular, such encryption is defined in terms of the values $r$, $r'$ and $r''$: $r$ and $r'$ are randomly chosen from $\mathbb{Z}_q$ in both games, whereas $r''$ is computed differently in the two games. For the sake of clarity, we will denote with $[r'']_2$ and $[r'']_3$ the value of such quantity in game $\mathbf{G}_{0,i}^2$ and $\mathbf{G}_{0,i}^3$, respectively. Notice that $[r'']_2$ is a linear combination of $r$, $r'$ (and other quantities), whereas $[r'']_3$ is uniformly distributed in $\mathbb{Z}_q$, independently of anything else.

Define

$$\boldsymbol{W} \doteq \left(\{c_j^t, r_j^t\}_{\substack{j\,=\,1\\ t\,\neq\,\hat{t}}}^{2v+2}, \{c_j^{\hat{t}}, s_j, r_j^{\hat{t}}\}_{j=1}^{i}, \{c_j^{\hat{t}}, r_j^{\hat{t}}\}_{j=i+1}^{2v+2}, r, r'\right)$$

and consider the quantity

$$\boldsymbol{V} \doteq (\mathsf{Coins}, w, \sigma^*, r^*, \boldsymbol{W})$$

where $\mathsf{Coins}$ represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, $\sigma^*$ is the random bit chosen by the challenger in step 8. of the attack game and $r^*$ is the randomness used to create the challenge $\psi^*$.

The remaining randomness used during the attack game consists of the $2v+2$ coefficients of the polynomials $A^0(\cdot)$, $B^0(\cdot)$ and can be represented by a vector $\boldsymbol{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$:

$$\boldsymbol{\alpha} \doteq (a_0, a_1, \ldots, a_v, b_0, b_1, \ldots, b_v)^T.$$

Consider the vector $\boldsymbol{\beta} \in \mathbb{Z}_q^{(2v+2)\times 1}$ defined as:

$$\boldsymbol{\beta} \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \mathbf{A}_1, \ldots, \mathbf{A}_v, r'')^T$$

where $\mathbf{X}_0 \doteq A^0(0) + wB^0(0)$, $\mathbf{X}_\ell \doteq A^0(\ell) + wB^0(\ell)$ and $\mathbf{A}_\ell \doteq A^0(x_\ell)$ for $\ell = 1, \ldots, v$, and $r'' \doteq \log_g u''$.

It is clear by inspection that all the information in the adversary's view is completely determined by $\boldsymbol{V}$ and $\boldsymbol{\beta}$. In particular, the initial public key $PK^0$ is fixed by $\boldsymbol{\beta}$ and $w$; the secret keys of the corrupted users are determined by the choice of $\boldsymbol{\beta}$, $\mathsf{Coins}$ and $w$; the "special" reset message is fixed by $PK^0$, $\mathsf{Coins}$, $r''$ and all the randomness in $\boldsymbol{W}$; and the resulting public key $PK^{\hat{t}}$ only depends on $PK^0$ and $\boldsymbol{W}$. Thus, if the distribution of $\boldsymbol{V}$ and $\boldsymbol{\beta}$ is the same in both games $\mathbf{G}_{0,i}^2$ and $\mathbf{G}_{0,i}^3$, it will follow that $\Pr[T_{0,i}^3] = \Pr[T_{0,i}^2]$. Since the definition of $r''$ is the only difference between game $\mathbf{G}_{0,i}^2$ and $\mathbf{G}_{0,i}^3$, and in $\mathbf{G}_{0,i}^3$ the value of $[r'']_3$ is chosen uniformly from $\mathbb{Z}_q$, independently of anything else, it suffices to show that the distribution of $[r'']_2$, conditioned on $\boldsymbol{V}$ and the first $2v+1$ entries of $\boldsymbol{\beta}$, is also uniform in $\mathbb{Z}_q$.

In game $\mathbf{G}_{0,i}^2$, the quantities in $\boldsymbol{V}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are related according to the following matrix equation:

$$[\boldsymbol{\beta}]_2 = \mathbf{M} \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}$$

where $[\boldsymbol{\beta}]_2$ denotes the value of $\boldsymbol{\beta}$ in game $\mathbf{G}_{0,i}^2$ (i.e. when the value of the last entry is $[r'']_2$), $\boldsymbol{\gamma} \in \mathbb{Z}_q^{(2v+2)\times 1}$ is the vector

$$\boldsymbol{\gamma} \doteq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ rD^{0,\hat{t}-1}(0) + wr'E^{0,\hat{t}-1}(0) + \log_g enc(c_{i+1}^{\hat{t}}) \end{pmatrix}$$

and $\mathbf{M} \in \mathbb{Z}_q^{(2v+2)\times(2v+2)}$ is the matrix

$$\mathbf{M} \doteq \begin{pmatrix} 1 & 0 & \dots & 0 & w & 0 & \dots & 0 \\ 1 & 1 & \dots & 1 & w & w & \dots & w \\ & & \vdots & & & & \vdots & \\ 1 & v & \dots & v^v & w & wv & \dots & wv^v \\ 1 & x_1 & \dots & x_1^v & 0 & 0 & \dots & 0 \\ & & \vdots & & & & \vdots & \\ 1 & x_v & \dots & x_v^v & 0 & 0 & \dots & 0 \\ r & 0 & \dots & 0 & wr' & 0 & \dots & 0 \end{pmatrix}$$

The above matrix describes all the constraints on $\boldsymbol{\alpha}$ arising from the information in the adversary's view in game $\mathbf{G}_{0,i}^2$ (which, as noted above, can be described just by $\boldsymbol{V}$ and $[\boldsymbol{\beta}]_2$). In other words, all other constraints on $\boldsymbol{\alpha}$ are linear combination of the above, possibly using coefficients from $\boldsymbol{V}$. In particular, the constraints that the adversary can derive from knowledge of the values $B^0(x_\ell)$, $\ell = 1, \dots, v$ (which come from the secret keys that $\mathcal{A}$ got via Join queries) can be obtained from the constraints corresponding to $\mathbf{X}_0$, $\mathbf{X}_1$, …, $\mathbf{X}_v$, $\mathbf{A}_1$, …, $\mathbf{A}_v$ and the value of $w$. As for Revoke queries, notice that the public key $PK$ resulting from invalidating the secret key of an arbitrary user $z$ during time period $t$, does not provide any new information about $\boldsymbol{\alpha}$ to the adversary. Indeed, $PK$ only differs from the previous public key in that it contains the value $h_z = g^{A^t(z)} g'^{B^t(z)}$ which is determined by the quantity:

$$\begin{aligned} \mathbf{X} &\doteq A^t(z) + wB^t(z) - (D^{0,t}(z) + wE^{0,t}(z)) \\ &= A^0(z) + wB^0(z). \end{aligned}$$

Since such value is just a point of the polynomial $v$-degree $A^0(\cdot) + wB^0(\cdot)$, which is completely fixed by the values $\mathbf{X}_0$, $\mathbf{X}_1$, …, $\mathbf{X}_v$, it immediately follows that the constraint on $\boldsymbol{\alpha}$ induced by $\mathbf{X}$ is a linear combination of the first $v+1$ rows of $\mathbf{M}$. Similarly, the $v$ values $u_1$, …, $u_v$ included in the $(v+1)$th ciphertext of the "special" reset message are determined by the quantities $\mathbf{X}_{z_1}, \dots, \mathbf{X}_{z_v}$, where, for $\ell = 1, \dots, v$, $\mathbf{X}_{z_\ell}$ is defined as:

$$rA^{\hat{t}-1}(z_\ell) + wr'B^{\hat{t}-1}(z_\ell) - (rD^{0,\hat{t}-1}(z_\ell) + wr'E^{0,\hat{t}-1}(z_\ell))$$

or equivalently

$$\mathbf{X}_{z_\ell} \doteq rA^0(z_\ell) + wr'B^0(z_\ell).$$

Such values are just points of the $v$-degree polynomial

$$rA^0(\cdot) + wr'B^0(\cdot)$$

which is determined by $\mathbf{A}_1$, …, $\mathbf{A}_v$, $B^0(x_1)$, …, $B^0(x_v)$, $r$, $r'$, $w$ and $[r'']_2$. Thus, it follows that all the constraints on $\boldsymbol{\alpha}$ induced $\mathbf{X}_{z_1}$, …, $\mathbf{X}_{z_v}$ by are linear combinations of the rows of $\mathbf{M}$.

Moreover, $\mathbf{M}$ has rank $(2v+2)$, provided that $r \neq r'$ and $w \neq 0$, since the corrupted users $x_1$, …, $x_v$ are assumed to be distinct.

As soon as we fix a value for $\boldsymbol{V}$, vector $\boldsymbol{\gamma}$ and the first $v+1$ rows of matrix $\mathbf{M}$ are completely fixed, but $\boldsymbol{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. If we additionally fix the value of the first $(v+1)$ components of $[\boldsymbol{\beta}]_2$, the initial public key $PK^0$ is fixed; it follows that the first identity $x_1$ that $\mathcal{A}$ chooses to corrupt is also fixed and thus the $(v+2)$th row of $\mathbf{M}$ is determined. Fixing also a value for $\mathbf{A}_1$ (which is the $(v+2)$th entry of $[\boldsymbol{\beta}]_2$), the value of $\mathbf{B}_1$ is fixed too, so that all the information on which the adversary can base her choice of $x_2$ is fixed, and thus the $(v+3)$th row of $\mathbf{M}$ is determined as well. By a similar reasoning, it follows that fixing the first $(v+i+1)$ entries of $[\boldsymbol{\beta}]_2$ determines the $(v+i+2)$th row of $\mathbf{M}$, for $i = 1, \dots, v$. Hence, by Lemma 1, we can conclude that the conditional distribution of $[r'']_2$, w.r.t. $\boldsymbol{V}$ and to all other components of $[\boldsymbol{\beta}]_2$, is also uniform over $\mathbb{Z}_q$, from which it follows that

$$\Pr[T_{0,i}^3] = \Pr[T_{0,i}^2]. \tag{13}$$

**Game $\mathbf{G}_{0,i}^4$.** Game $\mathbf{G}_{0,i}^4$ is defined to be identical to $\mathbf{G}_{0,i+1}$. Thus, $\mathbf{G}_{0,i}^4$ differs from $\mathbf{G}_{0,i}^3$ only in that the values $u$ and $u'$ in the $(i+1)$th ciphertext in the "special" $\mathsf{reset}$ message are consistent, rather than being nearly independent, as in game $\mathbf{G}_{0,i}^3$. Namely, the values $u$ and $u'$ are now computed as $u \doteq g^r$ and $u' \doteq g'^r$, for the same random $r \in \mathbb{Z}_q$. It follows that any difference in behavior between games $\mathbf{G}_{0,i}^3$ and $\mathbf{G}_{0,i}^4$ can be used to distinguish Diffie-Hellman tuples from totally random tuples. Hence,

$$\left|\Pr[T_{0,i}^4] - \Pr[T_{0,i}^3]\right| \leq \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{14}$$

Combining Equations (11), (12), (13) and (14) we get Equation (10), for all $i = 0, \ldots, 2v+1$; then, by definition of the hybrid sequence $\mathbf{G}_{0,0}, \ldots, \mathbf{G}_{0,2v+2}$, the thesis follows. □

**Lemma 3.** *$\left|\Pr[T_2] - \Pr[T_1]\right| \leq 2\ \mathsf{AdvDDH}_{\mathcal{G}}(k)$.*

*Proof.* Recall that $\mathbf{G}_2$ differs from $\mathbf{G}_1$ only in the way the challenge ciphertext $\psi^*$ is computed: in particular, in game $\mathbf{G}_1$, $\psi^*$ encrypts either one of the two messages $M_0$ and $M_1$ chosen by the adversary, whereas in $\mathbf{G}_2$, $\psi^*$ encrypts a totally random message $M \xleftarrow{r} \mathcal{G}$.

To reach the thesis, we consider the sequence of games $\mathbf{G}_1^0 \equiv \mathbf{G}_1$, $\mathbf{G}_1^1$, $\mathbf{G}_1^2$, $\mathbf{G}_1^3$, $\mathbf{G}_1^4 \equiv \mathbf{G}_2$, defined below.

**Game $\mathbf{G}_1^1$.** It operates as $\mathbf{G}_1^0$, except that the challenge ciphertext is computed as follows:

$$\langle u^*, u'^*, u''^*, \langle z_\ell^*, {u^*}^{A^{t^*}(z_\ell^*)} {u'^*}^{B^{t^*}(z_\ell^*)} \rangle_{\ell=1}^{v} \rangle$$

where $u^* \doteq g^{r^*}$, $u'^* \doteq g'^{r^*}$, $u''^* \doteq {u^*}^{A^{t^*}(0)} {u'^*}^{B^{t^*}(0)} \cdot M_{\sigma^*}$ and $r^* \xleftarrow{r} \mathbb{Z}_q$. This syntactic change does not affect the adversary's view, and thus

$$\Pr[T_1^1] = \Pr[T_1^0]. \tag{15}$$

**Game $\mathbf{G}_1^2$.** To turn game $\mathbf{G}_1^1$ into game $\mathbf{G}_1^2$ we make another change to the way in which the challenge ciphertext is computed. Namely, the value $u'^*$ is now computed as $u'^* \doteq g'^{r'^*}$, for a random $r'^* \in \mathbb{Z}_q$ such that $r'^* \neq r^*$. In other words, in game $\mathbf{G}_1^2$ the values $u^*$ and $u'^*$ are nearly independent (being subject only to $r^* \neq r'^*$), whereas in game $\mathbf{G}_1^1$ they are obtained using the same value $r^*$. Therefore, using a standard reduction argument, any difference in behavior between games $\mathbf{G}_1^1$ and $\mathbf{G}_1^2$ can be used to distinguish Diffie-Hellman tuples from totally random tuples. Hence,

$$\left|\Pr[T_1^2] - \Pr[T_1^1]\right| \leq \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{16}$$

**Game $\mathbf{G}_1^3$.** To define game $\mathbf{G}_1^3$, we again modify the challenge ciphertext: specifically, the value $u''^*$ is now computed as $g^{r''^*}$, for a random $r''^* \in \mathbb{Z}_q$.

To prove that $\Pr[T_1^3] = \Pr[T_1^2]$, we first consider all the quantities that can affect event $T_1^2$ in game $\mathbf{G}_1^2$ and event $T_1^3$ in game $\mathbf{G}_1^3$, and then we show that these quantities have the same joint distribution in both games.

Let $D^{t^*}(\cdot)$ and $E^{t^*}(\cdot)$ be the randomizing polynomials used in the last $\mathsf{New\text{-}period}$ operation before the challenge ciphertext was created. (If no $\mathsf{New\text{-}period}$ occurred at all during the attack game, then let both $D^{t^*}(\cdot)$ and $E^{t^*}(\cdot)$ be just the zero polynomial.)

Let $\bar{t}$ be the total number of $\mathsf{New\text{-}period}$ operations that occur during the entire game, and for $t = 1, \ldots, \bar{t}$, let $c_1^t, \ldots, c_{2v+2}^t$ be the coefficients of the randomizing polynomials $D^t(\cdot)$ and $E^t(\cdot)$ used in the $t$th $\mathsf{New\text{-}period}$ operation. For $t = 1, \ldots, \bar{t}$, and $j = 1, \ldots, 2v+2$, let $r_j^t$ be the randomness used to encrypt (the encoding of) coefficient $c_j^t$ in the $t$th $\mathsf{reset}$ message.

Observe that the challenge ciphertext $\psi^*$ is the only value in the adversary's view which is computed differently in game $\mathbf{G}_1^2$ and game $\mathbf{G}_1^3$. In particular, such encryption is defined in terms of the values $r^*$, $r'^*$ and $r''^*$: $r^*$ and $r'^*$ are randomly chosen from $\mathbb{Z}_q$ in both games, whereas $r''^*$ is computed differently in the two games. For the sake of clarity, we will denote with $[r''^*]_2$ and $[r''^*]_3$ the value of such quantity in game $\mathbf{G}_1^2$ and $\mathbf{G}_1^3$, respectively. Notice that $[r''^*]_2$ is a linear combination of $r^*$, $r'^*$ (and other quantities), whereas $[r''^*]_3$ is uniformly distributed in $\mathbb{Z}_q$, independently of anything else.

The rest of our analysis proceeds differently depending on whether the adversary's strategy caused the "special" New-period operation to occur or not. The case in which no New-period operation occurred at step 7. of the attack game is slightly simpler, so we discuss it first.

Case 1. Consider the quantity

$$\boldsymbol{V} \doteq (\mathsf{Coins}, w, \{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}}, \sigma^*, r^*, r'^*)$$

where Coins represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, and $\sigma^*$ is the random bit chosen by the challenger in step 8. of the attack game.

The remaining randomness used during the attack game consists of the $2v+2$ coefficients of the polynomials $A^0(\cdot)$, $B^0(\cdot)$ and can be represented by a vector $\boldsymbol{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$:

$$\boldsymbol{\alpha} \doteq (a_0, a_1, \ldots, a_v, b_0, b_1, \ldots, b_v)^T.$$

Consider the vector $\boldsymbol{\beta} \in \mathbb{Z}_q^{(2v+2)\times 1}$ defined as:

$$\boldsymbol{\beta} \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \mathbf{A}_1, \ldots, \mathbf{A}_v, r''^*)^T$$

where $\mathbf{X}_0 \doteq A^0(0) + wB^0(0)$, $\mathbf{X}_\ell \doteq A^0(\ell) + wB^0(\ell)$ and $\mathbf{A}_\ell \doteq A^0(x_\ell)$ for $\ell = 1, \ldots, v$, and $r''^* \doteq \log_g u''^*$.

It is clear by inspection that all the information in the adversary's view is completely determined by $\boldsymbol{V}$ and $\boldsymbol{\beta}$. Thus, if the distribution of $\boldsymbol{V}$ and $\boldsymbol{\beta}$ is the same in both games $\mathbf{G}_1^2$ and $\mathbf{G}_1^3$, it will follow that $\Pr[T_1^3] = \Pr[T_1^2]$. Since the definition of $r''^*$ is the only difference between game $\mathbf{G}_1^2$ and $\mathbf{G}_1^3$, and in $\mathbf{G}_1^3$ the value of $[r''^*]_3$ is chosen uniformly from $\mathbb{Z}_q$, independently of anything else, it suffices to show that the distribution of $[r''^*]_2$, conditioned on $\boldsymbol{V}$ and the first $2v+1$ entries of $\boldsymbol{\beta}$, is also uniform in $\mathbb{Z}_q$.

In game $\mathbf{G}_1^2$, the quantities in $\boldsymbol{V}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are related according to the following matrix equation:

$$[\boldsymbol{\beta}]_2 = \mathbf{M} \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}$$

where $[\boldsymbol{\beta}]_2$ denotes the value of $\boldsymbol{\beta}$ in game $\mathbf{G}_1^2$ (i.e. when the value of the last entry is $[r''^*]_2$), $\boldsymbol{\gamma} \in \mathbb{Z}_q^{(2v+2)\times 1}$ is the vector

$$\boldsymbol{\gamma} \doteq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ r^*(D^{0,t^*}(0)) + wr'^*(E^{0,t^*}(0)) + \log_g M_{\sigma^*} \end{pmatrix}$$

and $\mathbf{M} \in \mathbb{Z}_q^{(2v+2)\times(2v+2)}$ is the matrix

$$\mathbf{M} \doteq \begin{pmatrix} 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 1 & w & w & \ldots & w \\ & & \vdots & & & & \vdots & \\ 1 & v & \ldots & v^v & w & wv & \ldots & wv^v \\ 1 & x_1 & \ldots & x_1^v & 0 & 0 & \ldots & 0 \\ & & \vdots & & & & \vdots & \\ 1 & x_v & \ldots & x_v^v & 0 & 0 & \ldots & 0 \\ r^* & 0 & \ldots & 0 & wr'^* & 0 & \ldots & 0 \end{pmatrix}$$

The above matrix $\mathbf{M}$ is square, has full rank (provided that $r^* \neq r'^*$ and $w \neq 0$) and it describes all the constraints on the $(2v+2)$ unknowns represented by $\boldsymbol{\alpha}$, that can be derived from the information in the adversary's view in $\mathbf{G}_1^2$. In particular, the fact that no New-period operation occurred during execution of step 7. of the attack game guarantees that the identities included in the public key $PK^*$ that was used to create the challenge ciphertext $\psi^*$ are exactly those of the users corrupted by the adversary. Hence, the constraints on $\boldsymbol{\alpha}$ arising from the last $v$ components of the challenge ciphertext $\psi^*$ can be obtained as linear combination of the constraints specified by $\mathbf{M}$.

As soon as we fix a value for $\boldsymbol{V}$, the first $2v + 1$ entries of vector $\boldsymbol{\gamma}$ and the first $v + 1$ rows of matrix $\mathbf{M}$ are completely fixed, but $\boldsymbol{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. If we additionally fix the value of the first $(v + 1)$ components of $[\boldsymbol{\beta}]_2$, the initial public key $PK^0$ is fixed; it follows that the first identity $x_1$ that $\mathcal{A}$ chooses to corrupt is also fixed and thus the $(v + 2)$th row of $\mathbf{M}$ is determined. Fixing also a value for $A_1$ (which is the $(v + 2)$th entry of $[\boldsymbol{\beta}]_2$), the value of $B_1$ is fixed too, so that all the information on which the adversary can base her choice of $x_2$ is fixed, and thus the $(v + 3)$th row of $\mathbf{M}$ is determined as well. By a similar reasoning, it follows that fixing the first $(v + \ell + 1)$ entries of $[\boldsymbol{\beta}]_2$ determines the $(v + \ell + 2)$th row of $\mathbf{M}$, for $\ell = 1, \ldots, v$. In particular, fixing all the entries of the $[\boldsymbol{\beta}]_2$ but the last, fixes all the information that adversary $\mathcal{A}$ sees before asking for her challenge: thus, her choice of $M_0$, $M_1$ is determined, so that the last entry of $\boldsymbol{\gamma}$ is fixed, too. Hence, by Lemma 1, we can conclude that the conditional distribution of $[r''^*]_2$, w.r.t. $\boldsymbol{V}$ and to all the other components of $[\boldsymbol{\beta}]_2$, is also uniform over $\mathbb{Z}_q$, from which it follows that

$$\Pr[T_1^3] = \Pr[T_1^2]. \tag{17}$$

Case 2. We now discuss the case in which the "special" New-period operation takes place: in particular, let $D^{\hat{t}}(\cdot)$ and $E^{\hat{t}}(\cdot)$ be the randomizing polynomials used in such New-period operation. Consider the quantity

$$\boldsymbol{V} \doteq \left(\mathsf{Coins}, w, \{c_j^t, r_j^t\}_{\substack{j\,=\,1\\ t\,\neq\,\hat{t}}}^{2v+2}, \{s_j, r_j^{\hat{t}}\}_{j=1}^{2v+2}, \sigma^*, r^*, r'^*\right)$$

where Coins represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, $\sigma^*$ is the random bit chosen by the challenger in step 8. of the attack game, and $s_1, \ldots, s_{2v+2}$ are the random elements of $\mathcal{G}$ that are encrypted by the "special" New-period operation in place of the randomizing coefficients $d_0^{\hat{t}}$, $d_1^{\hat{t}}$, …, $d_v^{\hat{t}}$, and $e_0^{\hat{t}}$, $e_1^{\hat{t}}$, …, $e_v^{\hat{t}}$.

The remaining randomness used during the attack game consists of these randomizing coefficients, along with the $2v + 2$ coefficients of the polynomials $A^0(\cdot)$, $B^0(\cdot)$ and can be represented by a vector $\boldsymbol{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(4v+4)\times 1}$, given in Figure 1.

Consider the vector $\boldsymbol{\beta} \in \mathbb{Z}_q^{(4v+3)\times 1}$ defined in Figure 1: it is clear by inspection that all the information in the adversary's view is completely determined by $\boldsymbol{V}$ and $\boldsymbol{\beta}$. In particular, the initial public key $PK^0$ is fixed by $\boldsymbol{\beta}$ and $w$; the secret keys of the corrupted users are determined by the choice of $\boldsymbol{\beta}$, Coins and $w$; the "special" reset message is fixed by $PK^0$, Coins, and all the randomness in $\boldsymbol{V}$; the resulting public key $PK^{\hat{t}}$ only depends on $\boldsymbol{\beta}$ and $w$; and the adversary's choice of $M_0$ and $M_1$ is fixed by $\boldsymbol{V}$ and the first $4v + 2$ entries of $\boldsymbol{\beta}$.

Thus, if the distribution of $\boldsymbol{V}$ and $\boldsymbol{\beta}$ is the same in both games $\mathbf{G}_1^2$ and $\mathbf{G}_1^3$, it will follow that $\Pr[T_1^3] = \Pr[T_1^2]$. Since the definition of $r''^*$ is the only difference between game $\mathbf{G}_1^2$ and $\mathbf{G}_1^3$, and in $\mathbf{G}_1^3$ the value of $[r''^*]_3$ is chosen uniformly from $\mathbb{Z}_q$, independently of anything else, it suffices to show that the distribution of $[r''^*]_2$, conditioned on $\boldsymbol{V}$ and the first $4v + 2$ entries of $\boldsymbol{\beta}$, is also uniform in $\mathbb{Z}_q$.

In game $\mathbf{G}_1^2$, the quantities in $\boldsymbol{V}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are related according to the following matrix equation:

$$[\boldsymbol{\beta}]_2 = \mathbf{M} \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}$$

where $[\boldsymbol{\beta}]_2$ denotes the value of $\boldsymbol{\beta}$ in game $\mathbf{G}_1^2$ (i.e. when the value of the last entry is $[r''^*]_2$) and $\boldsymbol{\gamma} \in \mathbb{Z}_q^{(4v+3)\times 1}$ and $\mathbf{M} \in \mathbb{Z}_q^{(4v+3)\times(4v+4)}$ are defined in Figure 1.

The matrix $\mathbf{M}$ in Figure 1 describes all the constraints on the $(4v + 4)$ unknowns represented by $\boldsymbol{\alpha}$, that can be derived from the information in the adversary's view in game $\mathbf{G}_1^2$. Notice that $\mathbf{M}$ includes the constraints on $\boldsymbol{\alpha}$ arising from the last $v$ components of the challenge ciphertext $\psi^*$. Moreover, since we are assuming that the "special" New-period operation took place during the execution of step 7. of the attack game, and that the adversary never revokes the users she corrupts, the identities $z_1^*$, …, $z_v^*$ specified in the

$$\boldsymbol{\alpha} \doteq (a_0, a_1, \ldots, a_v, b_0, b_1, \ldots, b_v, d_0^{\hat{t}}, d_1^{\hat{t}}, \ldots, d_v^{\hat{t}}, e_0^{\hat{t}}, e_1^{\hat{t}}, \ldots, e_v^{\hat{t}})^T$$

$$\boldsymbol{\beta} \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \hat{\mathbf{X}}_0, \hat{\mathbf{X}}_1, \ldots, \hat{\mathbf{X}}_v, \mathbf{A}_1, \ldots, \mathbf{A}_v, \mathbf{X}_1^*, \ldots, \mathbf{X}_v^*, r''^*)^T$$

$$\boldsymbol{\gamma} \doteq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ D^{0,\hat{t}-1}(0) + wE^{0,\hat{t}-1}(0) \\ D^{0,\hat{t}-1}(1) + wE^{0,\hat{t}-1}(1) \\ \vdots \\ D^{0,\hat{t}-1}(v) + wE^{0,\hat{t}-1}(v) \\ 0 \\ \vdots \\ 0 \\ r^*(D^{0,\hat{t}-1}(z_1^*) + D^{\hat{t}+1,t^*}(z_1^*)) + wr'^*(E^{0,\hat{t}-1}(z_1^*) + E^{\hat{t}+1,t^*}(z_1^*)) \\ \vdots \\ r^*(D^{0,\hat{t}-1}(z_v^*) + D^{\hat{t}+1,t^*}(z_v^*)) + wr'^*(E^{0,\hat{t}-1}(z_v^*) + E^{\hat{t}+1,t^*}(z_v^*)) \\ r^*(D^{0,\hat{t}-1}(0) + D^{\hat{t}+1,t^*}(0)) + wr'^*(E^{0,\hat{t}-1}(0) + E^{\hat{t}+1,t^*}(0)) + \log_g M_{\sigma^*} \end{pmatrix}$$

$$\mathbf{M} \doteq \begin{pmatrix} 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 1 & w & w & \ldots & w & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ & & \vdots & & & & \vdots & & & & \vdots & & & & \vdots & \\ 1 & v & \ldots & v^v & w & wv & \ldots & wv^v & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 1 & w & w & \ldots & w & 1 & 1 & \ldots & 1 & w & w & \ldots & w \\ & & \vdots & & & & \vdots & & & & \vdots & & & & \vdots & \\ 1 & v & \ldots & v^v & w & wv & \ldots & wv^v & 1 & v & \ldots & v^v & w & wv & \ldots & wv^v \\ 1 & x_1 & \ldots & x_1^v & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ & & \vdots & & & & \vdots & & & & \vdots & & & & \vdots & \\ 1 & x_v & \ldots & x_v^v & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ r^* & r^*z_1^* & \ldots & r^*z_1^{*v} & wr'^* & wr'^*z_1^* & \ldots & wr'^*z_1^{*v} & r^* & r^*z_1^* & \ldots & r^*z_1^{*v} & wr'^* & wr'^*z_1^* & \ldots & wr'^*z_1^{*v} \\ & & \vdots & & & & \vdots & & & & \vdots & & & & \vdots & \\ r^* & r^*z_v^* & \ldots & r^*z_v^{*v} & wr'^* & wr'^*z_v^* & \ldots & wr'^*z_v^{*v} & r^* & r^*z_v^* & \ldots & r^*z_v^{*v} & wr'^* & wr'^*z_v^* & \ldots & wr'^*z_v^{*v} \\ r^* & 0 & \ldots & 0 & wr'^* & 0 & \ldots & 0 & r^* & 0 & \ldots & 0 & wr'^* & 0 & \ldots & 0 \end{pmatrix}$$

**Fig. 1.** Vectors $\boldsymbol{\alpha} \in \mathbb{Z}_q^{(4v+4)\times 1}$, $\boldsymbol{\beta} \in \mathbb{Z}_q^{(4v+3)\times 1}$ and $\boldsymbol{\gamma} \in \mathbb{Z}_q^{(4v+3)\times 1}$ and the matrix $\mathbf{M} \in \mathbb{Z}_q^{(4v+3)\times(4v+4)}$ used in the last information-theoretic argument of Lemma 3. Notation: $r''^* \doteq \log_g u''^*$, $\mathbf{X}_0 \doteq A^0(0) + wB^0(0)$, $\hat{\mathbf{X}}_0 \doteq A^{\hat{t}}(0) + wB^{\hat{t}}(0)$, and for $\ell = 1, \ldots, v$, $\mathbf{X}_\ell \doteq A^0(\ell) + wB^0(\ell)$, $\hat{\mathbf{X}}_\ell \doteq A^{\hat{t}}(\ell) + wB^{\hat{t}}(\ell)$, $\mathbf{A}_\ell \doteq A^0(x_\ell)$ and $\mathbf{X}_\ell^* \doteq r^*A^{t^*}(z_\ell^*) + wr'^*B^{t^*}(z_\ell^*)$.

public key $PK^{t^*}$ that is used to create the challenge ciphertext are all different from the identities $x_1$, …, $x_v$ of the corrupted users, so that $\mathbf{M}$ has full rank, provided that $r^* \neq r'^*$ and $w \neq 0$.

As soon as we fix a value for $\boldsymbol{V}$, the first $4v+2$ entries of vector $\boldsymbol{\gamma}$ and the first $2v+2$ rows of matrix $\mathbf{M}$ are completely fixed, but $\boldsymbol{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(4v+4)\times 1}$. If we additionally fix the value of the first $(2v+2)$ components of $[\boldsymbol{\beta}]_2$, the initial public key $PK^0$ is fixed (in fact, the public key $PK^{\hat{t}}$ resulting from the "special" New-period operation gets fixed, too); it follows that the first identity $x_1$ that $\mathcal{A}$ chooses to corrupt is also fixed and thus the $(2v+3)$th row of $\mathbf{M}$ is determined. Fixing also a value for $\mathbf{A}_1$ (which is the $(2v+3)$th entry of $[\boldsymbol{\beta}]_2$), the value of $\mathbf{B}_1$ is fixed too, so that all the information on which the adversary can base her choice of $x_2$ is fixed, and thus the $(2v+4)$th row of $\mathbf{M}$ is determined as well. By a similar reasoning, it follows that fixing the first $(2v+\ell+2)$ entries of $[\boldsymbol{\beta}]_2$ determines the $(2v+\ell+3)$th row of $\mathbf{M}$, for $\ell = 1, \ldots, v$. In particular, fixing the first $3v+2$ entries of $[\boldsymbol{\beta}]_2$ fixes all the information that adversary $\mathcal{A}$ sees before asking for her challenge: thus, the identities $z_1^*$, $\ldots$, $z_v^*$, as well as the two messages $M_0$, $M_1$ chosen by $\mathcal{A}$ are determined, so that all the remaining rows of $\mathbf{M}$, as well as the last entry of $\boldsymbol{\gamma}$ get fixed, too. Hence, by Lemma 1, we can conclude that the conditional distribution of $[r''^*]_2$, w.r.t. $\boldsymbol{V}$ and to all other components of $[\boldsymbol{\beta}]_2$, is uniform over $\mathbb{Z}_q$, from which it follows that Equation (17) holds in this case, too.

**Game $\mathbf{G}_1^4$.** Game $\mathbf{G}_1^4$ is defined to be identical to $\mathbf{G}_2$. Thus, $\mathbf{G}_1^4$ differs from $\mathbf{G}_1^3$ only in that the values $u^*$ and $u'^*$ in the challenge ciphertext $\psi^*$ are consistent, rather than being nearly independent, as in game $\mathbf{G}_1^3$. Namely, the values $u^*$ and $u'^*$ are now computed as $u^* \doteq g^{r^*}$ and $u'^* \doteq g'^{r^*}$, for the same random $r^* \in \mathbb{Z}_q$. It follows that any difference in behavior between games $\mathbf{G}_1^3$ and $\mathbf{G}_1^4$ can be used to distinguish Diffie-Hellman tuples from totally random tuples. Hence,

$$\left|\Pr[T_1^4] - \Pr[T_1^3]\right| \leq \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{18}$$

Combining Equations (15), (16), (17) and (18), the thesis follows. □

## 6 Dealing with 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" $\mathsf{D}$ which, as in previous work, is assumed to be stateless. In this section we present a formal model for traceability and two tracing algorithms that can be integrated within the scheme described in Section 4.

The first 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,[5] checks whether the suspected set includes all the user-keys that were used to generate the pirate device, and if so outputs the identity of one of the traitors.

The second method, a *non-black-box algorithm*, receives as input a "valid" key extracted from a pirate device which was constructed using the keys of at most $m$ users and deterministically recovers the identities of all the traitors.

### 6.1 Model for Traceability

The traceability adversary operates similarly to the revocation adversary described in Section 5. Namely, after receiving the initial public key of the system, adversary $\mathcal{A}$ can interleave (in any adaptively-chosen order) up to $m$ $\mathsf{Join}$ queries, upon which $\mathcal{A}$ receives the secret keys of the corresponding users (the *traitors*), and a polynomial number of $\mathsf{Revoke}$ queries. Notice that each $\mathsf{Revoke}$ will change the public key, and the adversary monitors these changes as well. Also notice that the final set of revoked users is likely very different, and typically disjoint from the set $\mathcal{T}$ of traitors. At the end, $\mathcal{A}$ outputs a pirate decoder $\mathsf{D}$ which presumably works well (in some sense more precisely clarified below), with the final public key $PK_{\mathcal{A}}$.

Formal Model for Traceability Adversary. We formalize the above attack scenario in terms of the traceability adversary attack game $\mathbf{G}_{\mathsf{trt}}^m(1^k)$, played between a challenger and the adversary.

1. Let $\langle PK, MSK\rangle := \mathsf{Setup}(1^k)$.
2. Let $L := 0$, $\mathcal{T} := \emptyset$.

[5] Recall, $m$ denotes the *collusion* threshold, and should not be confused with the *revocation* threshold $v$ defined in Section 4; e.g., in our schemes $m = \lfloor \frac{v}{2} \rfloor$.

3. Let $state := \langle L, PK, MSK, \mathcal{T} \rangle$.
4. $\mathsf{D} \leftarrow \mathcal{A}^{\mathsf{Join}(state,\cdot),\mathsf{Revoke}(state,\cdot)}(state.PK)$.
5. If $|\mathcal{T}| > m$ then exit.
6. Parse $state$ as $\langle L, PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathcal{T} \rangle$.
7. Output $\langle \mathsf{D}, PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathcal{T} \rangle$.

The definitions of the Join and Revoke oracles is the same as in Section 5.1, except that the role of the set Corr is now played by the set $\mathcal{T}$.

**Definition 8.** *For any public key $PK$, define the success probability of a decoder* D *as:*

$$\mathsf{Succ}_{PK}(\mathsf{D}) \doteq \Pr[M' = M \mid M \xleftarrow{r} \mathcal{G}; \psi^* \xleftarrow{r} \mathcal{E}(PK, M); \\ M' \xleftarrow{r} \mathsf{D}(\psi^*)]$$

*where the probability is over the random choice of $M$, the randomness used to create the challenge ciphertext $\psi^*$ and the coin tosses of* D.

Notice that the pirate decoder D expects to receive a ciphertext created under the public key $PK_{\mathcal{A}}$, but the quantity $\mathsf{Succ}_{PK}(\mathsf{D})$ can be defined for any public key anyway. Clearly, if D could notice the change, then it could stop working properly: in this case we can assume that it outputs a message $M' \neq M$.

The job of the tracing algorithm is to find one or all of the (at most) $m$ traitors whose keys were used by $\mathcal{A}$ in building D. The precise security guarantees depend on whether tracing is black-box or not. We describe both tracing methods in Section 6.2 and 6.3, respectively.

### 6.2 Black-Box Tracing

In the black-box model, the tracing algorithm is only allowed to query the pirate decoder D on a polynomial number of a random-looking ciphertexts, and from the plain observation of D's input/output behavior, the tracing algorithm should successfully in identifying (at least) one of the traitors.

This form of tracing is the most desirable, as it can be applied to any stateless pirate decoder. Similarly to previous work [3, 19, 21], though, the algorithm presented below only achieves a weaker variant of black-box tracing, called *black-box confirmation*. Informally, a black-box confirmation algorithm is a subroutine that tests 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 if so, it 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 universe $\mathcal{U}$ 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 space dramatically smaller; (2) all previous public-key traitor tracing schemes suffer from the same problem; (3) as observed in [14], the problem seems to be inherent to this setting.

However, we significantly improve upon 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 can be successfully applied to pirate decoders that work on at least an $\varepsilon$-fraction of correctly formed messages (rather than with probability 1), where $\varepsilon$ is the desired threshold below which the decoder is considered "useless" (following the "threshold tracing" approach of [18]).

**Definition 9 (Black-Box Confirmation Algorithm).**
*A* Black-Box Confirmation *(*BBC*) algorithm is a probabilistic, polynomial time oracle machine, taking as oracle input a pirate decoder* D, *and as regular input a public key $PK$, the corresponding master secret key $MSK$, and a set* Susp *of suspected traitors. Its output is either a user's identity $i$ or the special symbol ?.*

**Definition 10 ($\varepsilon$-Black-Box Confirmation Property).**
*Let $\mathcal{A}$ be any probabilistic, polynomial-time adversary, and let $\langle \mathsf{D}, PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathcal{T} \rangle$ be the output resulting from the adversary playing the traceability attack game $\boldsymbol{G}^m_{\mathsf{trt}}(1^k)$ with the challenger. A Black-Box Confirmation algorithm* BBC *satisfies* $\varepsilon$-Black-Box Confirmation *if for any* PPT *adversary $\mathcal{A}$ playing the $\boldsymbol{G}^m_{\mathsf{trt}}(1^k)$ game, the following two properties hold with all but negligible probability:*

- **Confirmation:** *whenever $\mathcal{T} \subseteq \mathsf{Susp}$, then the output of* $\mathsf{BBC}^{\mathsf{D}}(PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathsf{Susp})$ *is some identity $i \in \mathcal{T}$.*
- **Soundness:** *whenever* $\mathsf{BBC}^{\mathsf{D}}(PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathsf{Susp})$ *outputs $i \neq ?$, then $i \in \mathcal{T}$.*

**Black-Box Confirmation Algorithm** At a high level, our black-box confirmation algorithm BBC works as follows. Based on the current set $I$ of suspected users (initially set to Susp) and using the master secret key $MSK_{\mathcal{A}}$, it modifies the public key $PK_{\mathcal{A}}$ into a fake public key $PK(I)$. It then estimates the probability

$$\delta(I) \doteq \mathsf{Succ}_{PK(I)}(\mathsf{D})$$

by observing the behavior of D when fed with encryptions of the form $\mathcal{E}(PK(I), M)$, for random messages $M$. 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 $\varepsilon/m$). 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 fake public key $PK(I)$. Recall from Section 4 that the master secret key $MSK_{\mathcal{A}}$ consists of two random polynomials over $\mathbb{Z}_q^v[x]$. Let $\bar{t}$ be the total number of New-period operations that occur during the entire game, and for $t = 1, \ldots, \bar{t}$, let $c_1^t, \ldots, c_{2v+2}^t$ be the coefficients of the randomizing polynomials $D^t(\cdot)$ and $E^t(\cdot)$ used in the $t$th New-period operation. For $t = 1, \ldots, \bar{t}$, and $j = 1, \ldots, 2v+2$, let $r_j^t$ be the randomness used to encrypt (the encoding of) coefficient $c_j^t$ in the $t$th reset message. By Equation (8), $(A^{\bar{t}}(\cdot), B^{\bar{t}}(\cdot))$ denotes the master secret key corresponding to the public key $PK_{\mathcal{A}}$. Given the set $I$, we create two polynomials $A'(\cdot)$ and $B'(\cdot)$ uniformly distributed over $\mathbb{Z}_q^v[x]$ except they agree with $A^{\bar{t}}(\cdot)$ and $B^{\bar{t}}(\cdot)$ on points in $I$:

$$A'(x_s) = A^{\bar{t}}(x_s) \quad B'(x_s) = B^{\bar{t}}(x_s), \ \forall s \in I.$$

Notice that, since $|I| \leq m \leq v/2$, this creates no problem. We then create the public key $PK(I)$ as if the master secret key were $MSK' = (A'(\cdot), B'(\cdot))$ rather than $MSK_{\mathcal{A}} = (A^{\bar{t}}(\cdot), B^{\bar{t}}(\cdot))$. Specifically, we define

$$PK(I) \doteq (g, g', y', \langle z_\ell, h'_\ell \rangle_{\ell=1}^{v}).$$

where $y' \doteq g^{A'(0)} \cdot g'^{B'(0)}$, and $h'_\ell \doteq g^{A'(z_\ell)} \cdot g'^{B'(z_\ell)}$, for $\ell = 1, \ldots, v$.

**Correctness of Black-Box Tracing** The correctness of the black-box tracing algorithm described above follows from Theorem 2 and Theorem 3 stated below. Theorem 2 implies that if the decoder was useful at the start (i.e., $\mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D}) \geq \varepsilon$) and $\mathcal{T} \subseteq \mathsf{Susp}$, then the decoder cannot "notice" that $PK_{\mathcal{A}}$ was changed to $PK(\mathsf{Susp})$, i.e. $\delta(\mathsf{Susp}) \gtrsim \varepsilon$.[6] Coupled with the obvious fact that $\delta(\emptyset)$ 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 Theorem 3 below), $i$ must be one of the traitors. This shows the confirmation property.

**Theorem 2.** *Under the* DDH *assumption, if $\mathcal{T} \subseteq \mathsf{Susp}$ and $|\mathsf{Susp}| \leq v$, then $|\delta(\mathsf{Susp}) - \mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D})|$ is negligible.*

*Proof.* We again follow the structural approach of defining a sequence of "indistinguishable" games $\mathbf{G}_0, \mathbf{G}_1, \ldots$, all operating over the same underlying probability space. Each of these games consists of the BBC algorithm sending a ciphertext $\psi^*$ to the pirate decoder D; different games only differs in the way $\psi^*$ is computed. In the original game $\mathbf{G}_0$, the goal of the decoder D is to output a message $M'$ which is D's best guess at the random message $M$ encrypted within $\psi^*$; for each game $\mathbf{G}_j$, let $T_j$ be the event that $M = M'$ in $\mathbf{G}_j$.

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

**Game $\mathbf{G}_0$**: This game defines the probability $\mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D})$. In this game, the $\mathsf{BBC}$ algorithm takes as input the public key $PK_{\mathcal{A}}$, the corresponding master secret key $MSK_{\mathcal{A}}$ and a set $\mathsf{Susp}$ of suspected users; it then chooses a message $M \overset{r}{\leftarrow} \mathcal{G}$ and, using the public key $PK_{\mathcal{A}}$, encrypts it as follows:

$$
\begin{array}{ll}
E1. & r \overset{r}{\leftarrow} \mathbb{Z}_q \\
E2. & u \leftarrow g^r \\
E3. & u' \leftarrow g'^r \\
E4. & u'' \leftarrow M \cdot g^{A^{\bar{t}}(0)r} g'^{B^{\bar{t}}(0)r} \\
E5. & u_\ell \leftarrow g^{A^{\bar{t}}(z_\ell)r} g'^{B^{\bar{t}}(z_\ell)r}, \quad \ell = 1, \ldots, v \\
E6. & \psi^* \leftarrow \langle u, u', u'', \langle z_1, u_1 \rangle, \ldots, \langle z_v, u_v \rangle \rangle
\end{array}
$$

By definition, we have that

$$
\Pr[T_0] = \mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D}). \tag{19}
$$

**Game $\mathbf{G}_1$**: Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that in game $\mathbf{G}_1$ steps $E4$ and $E5$ are substituted with:

$$
\begin{array}{ll}
E4'. & u'' \leftarrow M \cdot u^{A^{\bar{t}}(0)} u'^{B^{\bar{t}}(0)} \\
E5'. & u_\ell \leftarrow u^{A^{\bar{t}}(z_\ell)} u'^{B^{\bar{t}}(z_\ell)}, \quad \ell = 1, \ldots, v
\end{array}
$$

Notice that the point of these changes is just to make explicit any functional dependency of $\psi^*$ on the quantities $u$ and $u'$. Since we just made a conceptual change, it clearly holds that

$$
\Pr[T_1] = \Pr[T_0]. \tag{20}
$$

**Game $\mathbf{G}_2$**: To define game $\mathbf{G}_2$, we make more changes to the encryption algorithm as follows:

$$
\begin{array}{ll}
E1'. & r \overset{r}{\leftarrow} \mathbb{Z}_q; \quad r' \overset{r}{\leftarrow} \mathbb{Z}_q \setminus \{r\} \\
E3'. & u' \leftarrow g'^{r'}
\end{array}
$$

Notice that while in game $\mathbf{G}_1$ the value $u$ and $u'$ are obtained using the same value $r$, in game $\mathbf{G}_2$ they are nearly independent, being subject only to $r \neq r'$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between games $\mathbf{G}_1$ and $\mathbf{G}_2$ can be used to construct a $\mathsf{PPT}$ adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$
\big| \Pr[T_2] - \Pr[T_1] \big| \leq \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{21}
$$

**Game $\mathbf{G}_3$**: To turn game $\mathbf{G}_2$ into game $\mathbf{G}_3$, we consider the set $\mathsf{Susp}$ and construct the public key $PK(\mathsf{Susp})$ as described above; specifically, two random polynomials $A'(\cdot)$ and $B'(\cdot)$ are chosen such that

$$
A'(x_s) = A^{\bar{t}}(x_s) \quad B'(x_s) = B^{\bar{t}}(x_s), \ \forall s \in \mathsf{Susp} \tag{22}
$$

and $PK(\mathsf{Susp})$ is set to be:

$$
PK(\mathsf{Susp}) \doteq \langle g, g', g^{A'(0)} g'^{B'(0)}, \langle z_\ell, g^{A'(z_\ell)} g'^{B'(z_\ell)} \rangle_{\ell=1}^{v} \rangle.
$$

Then, we change steps $E4'$ and $E5'$ of the encryption algorithm of game $\mathbf{G}_2$ as follows:

$$
\begin{array}{ll}
E4''. & u'' \leftarrow M \cdot u^{A'(0)} u'^{B'(0)} \\
E5''. & u_\ell \leftarrow u^{A'(z_\ell)} u'^{B'(z_\ell)}, \quad \ell = 1, \ldots, v
\end{array}
$$

Using the technique outlined in Section 5.2, in Lemma 4 below, we show that

$$\Pr[T_3] = \Pr[T_2]. \tag{23}$$

**Game $\mathbf{G}_4$**: In game $\mathbf{G}_4$, we "undo" the changes of game $\mathbf{G}_2$, restoring lines $E1$ and $E3$ of the encryption oracle to their original values:

$$\begin{aligned} &E1''. \quad r \xleftarrow{r} \mathbb{Z}_q \\ &E3''. \quad u' \leftarrow g'^r \end{aligned}$$

Notice that in game $\mathbf{G}_4$ the value $u$ and $u'$ are again obtained using the same value $r$, whereas in game $\mathbf{G}_3$ they are nearly independent, being subject only to $r \neq r'$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between games $\mathbf{G}_3$ and $\mathbf{G}_4$ can be used to construct a $\mathsf{PPT}$ adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\big|\Pr[T_4] - \Pr[T_3]\big| \leq \ \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{24}$$

Finally, observe that in $\mathbf{G}_4$ the encryption of the random message $M$ is obtained using the public key $PK(\mathsf{Susp})$: thus, game $\mathbf{G}_4$ is exactly the game which defines the probability $\delta(\mathsf{Susp})$ i.e.,

$$\Pr[T_4] = \delta(\mathsf{Susp}). \tag{25}$$

Combining Equations (19), (20), (21), (23), (24) and (25), we can conclude that $\mathcal{A}$ has only a negligible chance to tell whether the message $M$ was encrypted under the public keys $PK_{\mathcal{A}}$ or $PK(\mathsf{Susp})$; more precisely:

$$|\delta(\mathsf{Susp}) - \mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D})| \leq 2 \ \mathsf{AdvDDH}_{\mathcal{G}}(k).$$

□

**Lemma 4.** $\Pr[T_3] = \Pr[T_2]$

*Proof.* To prove the Lemma, we consider all the quantities that can affect event $T_2$ in game $\mathbf{G}_2$ and event $T_3$ in game $\mathbf{G}_3$, and then we show that these quantities are distributed according to the same joint distribution in both games.

Consider the quantity:

$$\boldsymbol{V} \doteq (\mathsf{Coins}_{\mathcal{A}}, \mathsf{Coins}_{\mathsf{D}}, w, M, r, r', \{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}})$$

where $\mathsf{Coins}_{\mathcal{A}}$ denotes the coin tosses of $\mathcal{A}$, $\mathsf{Coins}_{\mathsf{D}}$ denotes the coin tosses of $\mathsf{D}$, $w \doteq \log_g g'$, $M$ is the random message encrypted within $\psi^*$, $r$ and $r'$ are the random values used to create $\psi^*$, and

$$\{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}}$$

represents all the randomness used in the $\bar{t}$ $\mathsf{New\text{-}period}$ operations that took place during the $\mathbf{G}_{\mathsf{trt}}^{m}(1^k)$ attack game.

The remaining randomness used during games $\mathbf{G}_2$ and $\mathbf{G}_3$ consists of the $4v + 4$ coefficients of the polynomials $A^0(\cdot)$, $B^0(\cdot)$ (chosen by the $\mathsf{Setup}$ algorithm in step 1. of the $\mathbf{G}_{\mathsf{trt}}^{m}(1^k)$ attack game) and $A'(\cdot)$, $B'(\cdot)$ (used in game $\mathbf{G}_3$). This randomness can be represented with the vector

$$\boldsymbol{\alpha} \doteq (a_0, a_1, \ldots, a_v, b_0, b_1, \ldots, b_v)^T$$

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, and the vector

$$\boldsymbol{\alpha}' \doteq (a_0', a_1', \ldots, a_v', b_0', b_1', \ldots, b_v')^T$$

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, subject to the constraints arising from imposing Equation (22).

Let $\mathcal{T} = \{t_1, \ldots, t_m\}$ be the set of traitors and set

$$\mathbf{A}_j \doteq A^{\bar{t}}(x_{t_j}) \quad \mathbf{B}_j \doteq B^{\bar{t}}(x_{t_j}),\ j = 1, \ldots, m.$$

Notice that, since $\mathcal{T} \subseteq \mathsf{Susp}$, for $j = 1, \ldots, m$, it holds that $\mathbf{A}_j = A'(x_{t_j})$ and $\mathbf{B}_j = B'(x_{t_j})$.

Consider the quantity $\bar{\boldsymbol{\beta}} \in \mathbb{Z}_q^{(v+m+1)\times 1}$ defined as:

$$\bar{\boldsymbol{\beta}} \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \mathbf{A}_1, \ldots, \mathbf{A}_m)^T$$

where $\mathbf{X}_0 \doteq A^0(0) + wB^0(0)$, and $\mathbf{X}_\ell \doteq A^0(\ell) + wB^0(\ell)$, for $\ell = 1, \ldots, v$.

It is clear by inspection that all the information in the view of the adversary $\mathcal{A}$ during the attack game $\mathbf{G}^m_{\mathsf{trt}}(1^k)$ is completely determined by $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$. In particular, the initial public key $PK^0$ is fixed by $\bar{\boldsymbol{\beta}}$ and $w$, and the secret keys of the traitors are determined by the choice of $\bar{\boldsymbol{\beta}}$, $\mathsf{Coins}_{\mathcal{A}}$ and $w$.

Besides the information in $\mathcal{A}$'s view, which is completely determined by the value of $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, the only other quantity affecting $\mathsf{D}$'s behavior is the ciphertext $\psi^*$. This ciphertext is computed differently in games $\mathbf{G}_2$ and $\mathbf{G}_3$: for the sake of clarity, we will denote with $[\psi^*]_2$ and $[\psi^*]_3$ the value of such quantity in game $\mathbf{G}_2$ and $\mathbf{G}_3$, respectively. We now want to show that, conditioned on all the other information in $\mathsf{D}$'s view, $[\psi^*]_2$ and $[\psi^*]_3$ are distributed according to the same distribution in the two games.

In game $\mathbf{G}_2$, the ciphertext $[\psi^*]_2$ sent to the decoder is completely determined by $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and by the $v$-degree polynomial $X^{\bar{t}}(\cdot) \doteq rA^{\bar{t}}(\cdot) + wr'B^{\bar{t}}(\cdot)$. Similarly, in game $\mathbf{G}_3$, the ciphertext $[\psi^*]_3$ is completely determined by $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and by the $v$-degree polynomial $X'(\cdot) \doteq rA'(\cdot) + wr'B'(\cdot)$. Moreover, $[\psi^*]_2$ depends on $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and $X^{\bar{t}}(\cdot)$ according to the same functional dependence of $[\psi^*]_3$ upon $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and $X'(\cdot)$. Therefore, to prove the Lemma, it suffices to show that, conditioning on any fixed values of $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, $X^{\bar{t}}(\cdot)$ and $X'(\cdot)$ are distributed according to the same conditional probability distribution; namely, both are random polynomials over $\mathbb{Z}_q^v[x]$, subject to the constraint that their values at $x_{t_j}$ is $r\mathbf{A}_j + wr'\mathbf{B}_j$, for $j = 1, \ldots, m$.

By Lagrange interpolation, $X^{\bar{t}}(\cdot)$ can be identified with its value at the points 0, 1, …, $v - m$, $x_{t_1}$, …, $x_{t_m}$; define

$$\mathbf{X}^{\bar{t}}_\ell \doteq X^{\bar{t}}(\ell),\ \ell = 0, \ldots, v - m$$

and

$$\mathbf{X}^{\bar{t}}_{v-m+j} \doteq X^{\bar{t}}(x_{i_j}),\ j = 1, \ldots, m.$$

Similarly, we can also identify $X'(\cdot)$ with its value at the same $v + 1$ points; define

$$\mathbf{X}'_\ell \doteq X'(\ell),\ \ell = 0, \ldots, v - m$$

and

$$\mathbf{X}'_{v-m+j} \doteq X'(x_{t_j}),\ j = 1, \ldots, m.$$

As noticed above, the assumption that $\mathcal{T} \subseteq \mathsf{Susp}$ implies that for $j = 1, \ldots, m$:

$$A^{\bar{t}}(x_{t_j}) = A'(x_{t_j}) = \mathbf{A}_j, \quad B^{\bar{t}}(x_{t_j}) = B'(x_{t_j}) = \mathbf{B}_j.$$

Therefore, it follows that

$$\mathbf{X}_{v-m+j} = \mathbf{X}'_{v-m+j},\ j = 1, \ldots, m. \tag{26}$$

It only remains to be proven that, conditioning on fixed values of $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, the tuple $\mathbf{X}^{\bar{t}}_0$, …, $\mathbf{X}^{\bar{t}}_{v-m}$ and the tuple $\mathbf{X}'_0$, …, $\mathbf{X}'_{v-m}$ are distributed according to same joint conditional distribution. (Notice that fixing a value for $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, immediately fixes a value for the tuple $\mathbf{X}^{\bar{t}}_{v-m+j}$, $j = 1$, …, $m$, which by (26) is equal to $\mathbf{X}'_{v-m+j}$, $j = 1$, …, $m$.)

Recall that, in game $\mathbf{G}_3$, the polynomials $A'(\cdot)$ and $B'(\cdot)$ are chosen uniformly at random from $\mathbb{Z}_q^v[x]$, independently from anything else, but subject to the constraints in (22). Thus, the polynomial $X'(\cdot) = rA'(\cdot)$ $+$ $wr'B'(\cdot)$ is also random in $\mathbb{Z}_q^v[x]$, subject to the constraint that its value at $x_s$ is

$$rA^{\bar{t}}(x_s) + wr'B^{\bar{t}}(x_s),\ \forall s \in \mathsf{Susp}.$$

Therefore, conditioning on fixed values of $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, the tuple $\mathbf{X}'_0, \ldots, \mathbf{X}'_{v-m}$ is distributed uniformly at random in $\mathbb{Z}_q^{(v-m+1)\times 1}$. Hence, it suffices to show that, for $\ell = 0, \ldots, v-m$, the conditional distribution of $\mathbf{X}^{\bar{t}}_\ell$ w.r.t. $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and $\mathbf{X}^{\bar{t}}_0, \ldots, \mathbf{X}^{\bar{t}}_{\ell-1}$ is uniform over $\mathbb{Z}_q$. To this aim, fix $\ell \in \{0, \ldots, v-m\}$, and consider the following matrix equation:

$$\boldsymbol{\beta}_\ell = \mathbf{M}_\ell \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}_\ell$$

where $\boldsymbol{\beta}_\ell \in \mathbb{Z}_q^{(v+m+\ell+2)\times 1}$ is the vector

$$\boldsymbol{\beta}_\ell \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \mathbf{A}_1, \ldots, \mathbf{A}_m, \mathbf{X}^{\bar{t}}_0, \mathbf{X}^{\bar{t}}_1, \ldots, \mathbf{X}^{\bar{t}}_\ell)^T,$$

$\boldsymbol{\gamma}_\ell \in \mathbb{Z}_q^{(v+m+\ell+2)\times 1}$ is the vector

$$\boldsymbol{\gamma}_\ell \doteq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ D^{0,\bar{t}}(x_{t_1}) \\ \vdots \\ D^{0,\bar{t}}(x_{t_m}) \\ rD^{0,\bar{t}}(0) + wr'E^{0,\bar{t}}(0) \\ rD^{0,\bar{t}}(1) + wr'E^{0,\bar{t}}(1) \\ \vdots \\ rD^{0,\bar{t}}(\ell) + wr'E^{0,\bar{t}}(\ell) \end{pmatrix}$$

and $\mathbf{M}_\ell \in \mathbb{Z}_q^{(v+m+\ell+2)\times(2v+2)}$ is the matrix

$$\mathbf{M}_\ell \doteq \begin{pmatrix} 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 1 & w & w & \ldots & w \\ & & \vdots & & & & \vdots & \\ 1 & v & \ldots & v^v & w & wv & \ldots & wv^v \\ 1 & x_{t_1} & \ldots & x_{t_1}^v & 0 & 0 & \ldots & 0 \\ & & \vdots & & & & \vdots & \\ 1 & x_{t_m} & \ldots & x_{t_m}^v & 0 & 0 & \ldots & 0 \\ r & 0 & \ldots & 0 & wr' & 0 & \ldots & 0 \\ r & r & \ldots & r & wr' & wr' & \ldots & wr' \\ & & \vdots & & & & \vdots & \\ r & r\ell & \ldots & r\ell^v & wr' & wr'\ell & \ldots & wr'\ell^v \end{pmatrix}$$

By inspection, it is possible to see that the rows of matrix $\mathbf{M}_\ell$ are linearly independent, provided that $r \neq r'$ and $w \neq 0$: thus, the rank of $\mathbf{M}_\ell$ is $v+m+\ell+2$. As soon as we fix $\boldsymbol{V}$, vector $\boldsymbol{\gamma}_\ell$ and the first $v+1$ rows of $\mathbf{M}_\ell$ are determined, but $\boldsymbol{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. Similarly to the proof of Lemma 3, it is also possible to show that fixing the first $v+j$ entries of $\bar{\boldsymbol{\beta}}$ determines the $(v+j+1)$th row of $\mathbf{M}_\ell$, for $j = 1, \ldots, m$; and that moreover, fixing the first $v+m+1$ entries of $\bar{\boldsymbol{\beta}}$ determines all the remaining rows of $\mathbf{M}_\ell$.

Hence, by Lemma 1, we can conclude that the conditional distribution of $\mathbf{X}^{\bar{t}}_\ell$ w.r.t. $(\boldsymbol{V}, \bar{\boldsymbol{\beta}}, \mathbf{X}^{\bar{t}}_0, \ldots, \mathbf{X}^{\bar{t}}_{\ell-1})$ is uniform over $\mathbb{Z}_q$, $\forall \ell \in \{0, \ldots, v-m\}$. In other words, the value of $X^{\bar{t}}(\cdot)$ at any point is uniformly random, subject to the constraint

$$\mathbf{X}^{\bar{t}}(x_{t_j}) = r\mathbf{A}_j + wr'\mathbf{B}_j, \ \forall t_j \in \mathcal{T}.$$

Thus, $(\boldsymbol{V}, \bar{\boldsymbol{\beta}}, X^{\bar{t}}(\cdot)))$ has the same joint distribution as $(\boldsymbol{V}, \bar{\boldsymbol{\beta}}, X'(\cdot)$, completing the proof. □

We now move on to prove the soundness of the BBC algorithm, showing that it can accuse an innocent user with at most negligible probability. Informally this is true because, under the DDH assumption it is impossible to notice if the values $A'(x_i)$ and $B'(x_i)$ (which are unknown to the adversary since $i$ is assumed to be honest), were replaced by random noise $A''(x_i)$ and $B''(x_i)$. Thus, the behavior of the decoder will be the same regardless of whether $PK(I)$ or $PK(I \backslash \{i\})$ was used to encrypt the ciphertext. Since our algorithm only accuses a user $i$ when a sensible change occurs in the decryption capability of the pirate decoder, it follows that an innocent user will be blamed with at most negligible probability.

**Theorem 3.** *Under the* DDH *assumption, if* $|I| \leq v$ *and* $i \notin \mathcal{T}$*, then* $|\delta(I) - \delta(I \setminus \{i\})|$ *is negligible.*

*Proof.* Proceeding as in the proof of Theorem 2, we define a sequence of "indistinguishable" games $\mathbf{G}_0$, $\mathbf{G}_1$, ...: for each game $\mathbf{G}_j$, let $T_j$ be the event that decoder D correctly decrypts the challenge sent by the BBC algorithm in game $\mathbf{G}_j$.

**Game $\mathbf{G}_0$**: This game describes the experiment which defines the value of $\delta(I)$. In this game, the decoder D is fed with ciphertexts obtained encrypting random messages under the fake public key $PK(I)$, defined as:

$$PK(I) = \langle g, g', g^{A'(0)} g'^{B'(0)}, \langle z_\ell, g^{A'(z_\ell)} g'^{B'(z_\ell)} \rangle_{\ell=1}^{v} \rangle$$

where $A'(\cdot)$ and $B'(\cdot)$ are random $v$-degree polynomials subject to:

$$A'(x_s) = A^{\bar{t}}(x_s) \quad B'(x_s) = B^{\bar{t}}(x_s), \; \forall s \in I. \tag{27}$$

More precisely, the BBC algorithm chooses a random message $M$ and encrypts it as follows:

$$\begin{array}{ll} E1. & r \xleftarrow{r} \mathbb{Z}_q \\ E2. & u \leftarrow g^r \\ E3. & u' \leftarrow g'^r \\ E4. & u'' \leftarrow M \cdot g^{A'(0)r} g'^{B'(0)r} \\ E5. & u_\ell \leftarrow g^{A'(z_\ell)r} g'^{B'(z_\ell)r}, \quad \ell = 1, \ldots, v \\ E6. & \psi^* \leftarrow \langle u, u', u'', \langle z_1, u_1 \rangle, \ldots, \langle z_v, u_v \rangle \rangle \end{array}$$

By definition, we have that:

$$\Pr[T_0] = \delta(I). \tag{28}$$

**Game $\mathbf{G}_1$**: Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that in game $\mathbf{G}_1$ steps $E4$ and $E5$ are substituted with:

$$\begin{array}{ll} E4'. & u'' \leftarrow M \cdot u^{A'(0)} u'^{B'(0)} \\ E5'. & u_\ell \leftarrow u^{A'(z_\ell)} u'^{B'(z_\ell)}, \quad \ell = 1, \ldots, v \end{array}$$

Notice that the point of these changes is just to make explicit any functional dependency of $\psi^*$ on the quantities $u$ and $u'$. Since we just made a conceptual change, it clearly holds that

$$\Pr[T_1] = \Pr[T_0]. \tag{29}$$

**Game $\mathbf{G}_2$**: Game $\mathbf{G}_2$ is identical to game $\mathbf{G}_1$, except that in game $\mathbf{G}_2$ steps $E1$ and $E3$ are substituted with:

$$\begin{array}{ll} E1'. & r \xleftarrow{r} \mathbb{Z}_q; \quad r' \xleftarrow{r} \mathbb{Z}_q \setminus \{r\} \\ E3'. & u' \leftarrow g'^{r'} \end{array}$$

Notice that while in game $\mathbf{G}_1$ the value $u$ and $u'$ are obtained using the same value $r$, in game $\mathbf{G}_2$ they are nearly independent, being subject only to $r \neq r'$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between games $\mathbf{G}_1$ and $\mathbf{G}_2$ can be used to construct a $\mathsf{PPT}$ adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\big|\Pr[T_2] - \Pr[T_1]\big| \leq \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{30}$$

**Game $\mathbf{G}_3$**: To turn game $\mathbf{G}_2$ into game $\mathbf{G}_3$, we consider the set $I \setminus \{i\}$ and construct the public key $PK(I \setminus \{i\})$: two new random $v$-degree polynomials $A''(\cdot)$ and $B''(\cdot)$ are chosen such that

$$A''(x_s) = A^{\bar{t}}(x_s) \quad B''(x_s) = B^{\bar{t}}(x_s),\ \forall s \in I \setminus \{i\} \tag{31}$$

and $PK(I \setminus \{i\})$ is set to be:

$$PK(I \setminus \{i\}) \doteq \langle g, g', g^{A''(0)} g'^{B''(0)}, \langle z_\ell, g^{A''(z_\ell)} g'^{B''(z_\ell)} \rangle_{\ell=1}^{v} \rangle.$$

Notice that, for $s \in I \setminus \{i\}$, it holds that $A''(x_s) = A'(x_s)$ and $B''(x_s) = B'(x_s)$.

Finally, we change steps $E4'$ and $E5'$ of the encryption algorithm as follows:

$$\begin{aligned} &E4''. \quad u'' \leftarrow M \cdot u^{A''(0)} u'^{B''(0)} \\ &E5''. \quad u_\ell \leftarrow u^{A''(z_\ell)} u'^{B''(z_\ell)}, \quad \ell = 1, \ldots, v \end{aligned}$$

Using the technique described in Section 5.2, in Lemma 5 below, we show that

$$\Pr[T_3] = \Pr[T_2]. \tag{32}$$

**Game $\mathbf{G}_4$**: In game $\mathbf{G}_4$, we “undo” the changes of game $\mathbf{G}_2$, restoring lines $E1$ and $E3$ of the encryption oracle to their original values:

$$\begin{aligned} &E1''. \quad r \xleftarrow{r} \mathbb{Z}_q \\ &E3''. \quad u' \leftarrow g'^{r} \end{aligned}$$

Notice that in game $\mathbf{G}_4$ the value $u$ and $u'$ are again obtained using the same value $r$, whereas in game $\mathbf{G}_3$ they are nearly independent, being subject only to $r \neq r'$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between games $\mathbf{G}_3$ and $\mathbf{G}_4$ can be used to construct a $\mathsf{PPT}$ adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\big|\Pr[T_4] - \Pr[T_3]\big| \leq \mathsf{AdvDDH}_{\mathcal{G}}(k). \tag{33}$$

In game $\mathbf{G}_4$, the encryption of the random message $M$ is obtained using the public key $PK(I \setminus \{i\})$: thus, game $\mathbf{G}_4$ is exactly the game defining $\delta(I \setminus \{i\})$ i.e.,

$$\Pr[T_4] = \delta(I \setminus \{i\}). \tag{34}$$

By Equations (28), (29), (30), (32), (33) and (34), we can conclude that the adversary has only a negligible chance to tell whether the message $M$ was encrypted under $PK(I)$ or $PK(I \setminus \{i\})$; more precisely:

$$|\delta(I) - \delta(I \setminus \{i\})| \leq 2\ \mathsf{AdvDDH}_{\mathcal{G}}(k).$$

□

**Lemma 5.** $\Pr[T_3] = \Pr[T_2]$

*Proof.* To prove the Lemma, we consider all the quantities that can affect event $T_2$ in game $\mathbf{G}_2$ and event $T_3$ in game $\mathbf{G}_3$, and then we show that these quantities are distributed according to the same joint distribution in both games.

Let $\bar{m} \doteq |\mathcal{T} \cap I|$, where $\mathcal{T} = \{t_1, \dots, t_m\}$ is the set of traitors; w.l.o.g. assume that $\mathcal{T} \cap I = \{t_1, \dots, t_{\bar{m}}\}$. Also, set

$$\mathbf{A}_j \doteq A^{\bar{t}}(x_{t_j}) \quad \mathbf{B}_j \doteq B^{\bar{t}}(x_{t_j}),\ j = 1, \dots, m.$$

Notice that, since $i \notin \mathcal{T}$, for $1 \leq j \leq \bar{m}$ it also holds that:

$$\mathbf{A}_j = A'(x_{t_j}) = A''(x_{t_j}) \quad \mathbf{B}_j = B'(x_{t_j}) = B''(x_{t_j}).$$

Consider the quantity:

$$\boldsymbol{V} \doteq (\mathsf{Coins}_{\mathcal{A}}, \mathsf{Coins}_{\mathsf{D}}, w, M, r, r', \{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}})$$

where $\mathsf{Coins}_{\mathcal{A}}$ denotes the coin tosses of $\mathcal{A}$, $\mathsf{Coins}_{\mathsf{D}}$ denotes the coin tosses of $\mathsf{D}$, $w \doteq \log_g g'$, $\mathbf{X}_\ell \doteq (A^0(\ell) + B^0(\ell))$ for $\ell = 0, \dots, v$, $M$ is the random message encrypted within $\psi^*$, $r$ and $r'$ are the random values used to create $\psi^*$, and

$$\{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}}$$

represents all the randomness used in the $\bar{t}$ $\mathsf{New\text{-}period}$ operations that took place during the attack game $\mathbf{G}^{m}_{\mathsf{trt}}(1^k)$.

The remaining randomness used during games $\mathbf{G}_2$ and $\mathbf{G}_3$ consists of the $6v + 6$ coefficients of the polynomials $A^0(\cdot)$, $B^0(\cdot)$ (chosen by the $\mathsf{Setup}$ algorithm in step 1. of the $\mathbf{G}^{m}_{\mathsf{trt}}(1^k)$ attack game), $A'(\cdot)$, $B'(\cdot)$ (used in game $\mathbf{G}_2$), and $A''(\cdot)$, $B''(\cdot)$ (used in game $\mathbf{G}_3$). This randomness can be represented with the vector

$$\boldsymbol{\alpha} \doteq (a_0, a_1, \dots, a_v, b_0, b_1, \dots, b_v)^T$$

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, the vector

$$\boldsymbol{\alpha}' \doteq (a'_0, a'_1, \dots, a'_v, b'_0, b'_1, \dots, b'_v)^T$$

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, subject to the constraints arising from imposing Equation (27), and the vector

$$\boldsymbol{\alpha}'' \doteq (a''_0, a''_1, \dots, a''_v, b''_0, b''_1, \dots, b''_v)^T$$

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, subject to the constraints arising from imposing Equation (31).

Consider the quantity $\bar{\boldsymbol{\beta}} \in \mathbb{Z}_q^{(v+\bar{m}+1)\times 1}$ defined as:

$$\bar{\boldsymbol{\beta}} \doteq (\mathbf{X}_0, \mathbf{X}_1, \dots, \mathbf{X}_v, \mathbf{A}_1, \dots, \mathbf{A}_{\bar{m}})^T$$

where $\mathbf{X}_0 \doteq A^0(0) + wB^0(0)$, and $\mathbf{X}_\ell \doteq A^0(\ell) + wB^0(\ell)$, for $\ell = 1, \dots, v$.

It is clear by inspection that all the information in the view of the adversary $\mathcal{A}$ during the attack game $\mathbf{G}^{m}_{\mathsf{trt}}(1^k)$ is completely determined by $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$. In particular, the initial public key $PK^0$ is fixed by $\boldsymbol{V}$, and the secret keys of the traitors are determined by the choice of $\bar{\boldsymbol{\beta}}$, $\mathsf{Coins}_{\mathcal{A}}$ and $w$.

Besides the information in $\mathcal{A}$'s view, the only other quantity affecting $\mathsf{D}$'s behavior is the ciphertext $\psi^*$. This ciphertext is computed differently in games $\mathbf{G}_2$ and $\mathbf{G}_3$: for the sake of clarity, we will denote with $[\psi^*]_2$ and $[\psi^*]_3$ the value of such quantity in game $\mathbf{G}_2$ and $\mathbf{G}_3$, respectively. We now want to show that, conditioned on all the other information in $\mathsf{D}$'s view, $[\psi^*]_2$ and $[\psi^*]_3$ are distributed according to the same distribution in the two games.

In game $\mathbf{G}_2$, the ciphertext $[\psi^*]_2$ sent to the decoder is completely determined by $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and by the $v$-degree polynomial $X' \doteq rA'(\cdot) + wr'B'(\cdot)$. Similarly, in game $\mathbf{G}_3$, the ciphertext $[\psi^*]_3$ is completely determined by $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and by the $v$-degree polynomial $X''(\cdot) \doteq rA''(\cdot) + wr'B''(\cdot)$. Moreover, $[\psi^*]_2$ depends on $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and $X'(\cdot)$ according to the same functional dependence of $[\psi^*]_3$ upon $\boldsymbol{V}$, $\bar{\boldsymbol{\beta}}$ and $X''(\cdot)$. Therefore, to prove the

Lemma, it suffices to show that, conditioning on any fixed values of $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, $X'(\cdot)$ and $X''(\cdot)$ are distributed according to the same conditional probability distribution.

Recall that, in game $\mathbf{G}_2$, the polynomials $A'(\cdot)$ and $B'(\cdot)$ are chosen uniformly at random from $\mathbb{Z}_q^v[x]$, independently from anything else, but subject to the constraints in (27). Thus, the polynomial $X'(\cdot) = rA'(\cdot)$ $+\ wr'B'(\cdot)$ is also random in $\mathbb{Z}_q^v[x]$, subject to the constraint that its value at $x_s$ is

$$rA^{\bar{t}}(x_s) + wr'B^{\bar{t}}(x_s),\ \forall s \in I.$$

Similarly, in game $\mathbf{G}_3$, the polynomials $A''(\cdot)$ and $B''(\cdot)$ are chosen uniformly at random from $\mathbb{Z}_q^v[x]$, independently from anything else, but subject to the constraints in (31). Thus, the polynomial $X''(\cdot) = rA''(\cdot) + wr'B''(\cdot)$ is also random in $\mathbb{Z}_q^v[x]$, subject to the constraint that its value at $x_s$ is

$$rA^{\bar{t}}(x_s) + wr'B^{\bar{t}}(x_s),\ \forall s \in I \setminus \{i\}.$$

In other words, the distributions of $X'(\cdot)$ and $X''(\cdot)$ only differ in that the value of $X'(\cdot)$ at $x_i$ is fixed to be

$$\mathbf{X}'_i \doteq rA^{\bar{t}}(x_i) + wr'B^{\bar{t}}(x_i)$$

whereas the value of $X''(\cdot)$ at $x_i$ is a random element in $\mathbb{Z}_q$. Thus, to prove that $X'(\cdot)$ and $X''(\cdot)$ have the same conditional probability distribution w.r.t. $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, it suffices to show that, conditioning on any fixed values of $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$, the value $\mathbf{X}'_i$ is distributed uniformly at random in $\mathbb{Z}_q$.

To this aim, consider the following matrix equation:

$$\boldsymbol{\beta} = \mathbf{M} \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}$$

where $\boldsymbol{\beta} \in \mathbb{Z}_q^{(v+\bar{m}+2)\times 1}$ is the vector

$$\boldsymbol{\beta} \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \mathbf{A}_1, \ldots, \mathbf{A}_{\bar{m}}, \mathbf{X}'_i)^T,$$

$\boldsymbol{\gamma} \in \mathbb{Z}_q^{(v+\bar{m}+2)\times 1}$ is the vector

$$\boldsymbol{\gamma} \doteq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ D^{0,\bar{t}}(x_{t_1}) \\ \vdots \\ D^{0,\bar{t}}(x_{t_{\bar{m}}}) \\ rD^{0,\bar{t}}(x_i) + wr'E^{0,\bar{t}}(x_i) \end{pmatrix}$$

and $\mathbf{M} \in \mathbb{Z}_q^{(v+\bar{m}+2)\times(2v+2)}$ is the matrix

$$\mathbf{M} \doteq \begin{pmatrix} 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 1 & w & w & \ldots & w \\ & & \vdots & & & & \vdots & \\ 1 & v & \ldots & v^v & w & wv & \ldots & wv^v \\ 1 & x_{t_1} & \ldots & x_{t_1}^v & 0 & 0 & \ldots & 0 \\ & & \vdots & & & & \vdots & \\ 1 & x_{t_m} & \ldots & x_{t_{\bar{m}}}^v & 0 & 0 & \ldots & 0 \\ r & rx_i & \ldots & rx_i^v & wr' & wr'x_i & \ldots & wr'x_i^v \end{pmatrix}$$

By inspection, it is possible to see that the rows of matrix $\mathbf{M}$ are linearly independent, provided that $r \neq r'$ and $w \neq 0$: thus, the rank of $\mathbf{M}$ is $v + \bar{m} + 2$. As soon as we fix $\boldsymbol{V}$, vector $\boldsymbol{\gamma}$ and the first $v + 1$ rows of

$\mathbf{M}$ are determined, but $\boldsymbol{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. Similarly to the proof of Lemma 3, it is also possible to show that fixing the first $v+j$ entries of $\bar{\boldsymbol{\beta}}$ determines the $(v+j+1)$th row of $\mathbf{M}$, for $j = 1, \ldots, \bar{m}$; and that moreover, fixing the first $v+\bar{m}+1$ entries of $\bar{\boldsymbol{\beta}}$ determines all the remaining rows of $\mathbf{M}$.

Hence, by Lemma 1, we can conclude that the conditional distribution of $\mathbf{X}'_i$ w.r.t. $\boldsymbol{V}$ and $\bar{\boldsymbol{\beta}}$ is uniform over $\mathbb{Z}_q$. In other words, conditioning on the information seen by the adversary before receiving the challenge $\psi^*$, the value of $X'(\cdot)$ at $x_i$ looks random over $\mathbb{Z}_q$. It follows that $(\boldsymbol{V}, \bar{\boldsymbol{\beta}}, X'(\cdot))$ has the same joint distribution as $(\boldsymbol{V}, \bar{\boldsymbol{\beta}}, X''(\cdot))$, completing the proof. □

### 6.3 Non-Black-Box Tracing

In Section 6.3 we describe a non-black-box tracing algorithm which builds on the results of [3, 19], but it is tailored to our family of representations. Then, in Section 6.3, we analyze its security properties in the formal model for traceability of Section 6.1, under a non-black-box assumption, given below as Assumption 3. Before that, however, we develop some notation.

Notation. Recall that in the scheme of Section 4, the secret key of user $x_i$ consists of two points $A(x_i), B(x_i)$, which can be combined with the system's public key to obtain two leap vectors to be used in the decryption algorithm. More precisely, given the current public key

$$PK \doteq \langle g, g', y, \langle z_1, h_1\rangle, \ldots, \langle z_v, h_v\rangle\rangle,$$

it is possible to construct (by Definition 6) two leap vectors

$$\boldsymbol{\nu}_{A,i} \doteq \boldsymbol{\nu}^{x_i, A}_{z_1,\ldots,z_v} \quad \boldsymbol{\nu}_{B,i} \doteq \boldsymbol{\nu}^{x_i, B}_{z_1,\ldots,z_v}$$

where $(A(\cdot), B(\cdot))$ is the master secret key corresponding to the current public key $PK$. By Equations (2) and (4), $\boldsymbol{\nu}_{A,i}$ and $\boldsymbol{\nu}_{B,i}$ agree on the last $v$ components; thus, under the current public key $PK$, user $x_i$'s secret key can be compactly rewritten as

$$\begin{aligned}\boldsymbol{\delta}_i &\doteq \langle (\nu_{A,i})_0, (\nu_{B,i})_0, \boldsymbol{\delta}'_i\rangle \\ &\doteq \langle \lambda_0^{(i)} A(x_i), \lambda_0^{(i)} B(x_i), \langle \lambda_1^{(i)}, \ldots, \lambda_v^{(i)}\rangle\rangle,\end{aligned}$$

where $\lambda_0^{(i)}$, $\lambda_1^{(i)}, \ldots, \lambda_v^{(i)}$ are the Lagrange coefficients defined in Equations (3) and (4); recall that, for notational convenience, we use superscript $(i)$ to make explicit that a given set of Lagrange coefficients is relative to user $x_i$.

Notice that such vector $\boldsymbol{\delta}_i$ is a representation of $y$ w.r.t. $g$, $g'$, $h_1$, …, $h_v$; for short, when this is the case, in the following we will just say that $\boldsymbol{\delta}_i$ is a valid representation of the public key $PK$. Also notice that *any* such valid representation $\boldsymbol{\delta}$ of the current public key $PK$ would work for decrypting messages encrypted with $PK$; for a generic valid representation

$$\boldsymbol{\delta} \doteq \langle \gamma_a, \gamma_b, \gamma_1, \ldots, \gamma_v\rangle,$$

we will denote with $\boldsymbol{\delta}'$ its last $v$ entries:

$$\boldsymbol{\delta}' \doteq \langle \gamma_1, \ldots, \gamma_v\rangle.$$

In the non-black-box model, the tracing algorithm is assumed to be able of inspecting the content of a successful pirate decoder, and to extract the secret key hidden within it. More precisely, in designing and analyzing our non-black-box tracing algorithm, we make the following assumption:

**Assumption 3 (Non-Black-Box Assumption)**
*Let $\mathcal{A}$ be any probabilistic, polynomial-time adversary, and let $\langle \mathsf{D}, PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathcal{T}\rangle$ be the output resulting from the adversary playing the traceability attack game $\boldsymbol{G}^m_{\mathsf{trt}}(1^k)$ with the challenger. If $\mathsf{D}$ can correctly decrypt random ciphertexts encrypted using $PK_{\mathcal{A}}$ (in other words, $\mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D}) = 1$), then $\mathsf{D}$ contains a valid representation $\boldsymbol{\delta}$ of $PK_{\mathcal{A}}$, and it is possible to reverse-engineer $\mathsf{D}$ and extract $\boldsymbol{\delta}$.*

Assumption 3 is partially supported by Proposition 1 and it is essentially equivalent to what was previously assumed in [3]. It is also *a priori* much less restrictive than the non-black-box assumption made in [19], where the non-black-box analysis is subject to the hypothesis that the illegal key extracted from the pirate decoder is a convex linear combination of some of the traitors' keys. In fact, in Lemma 6 (whose proof is given in Section 6.3) we show that in our context, the seemingly more restrictive assumption from [19] actually follows from Assumption 3 and Assumption 2.

**Lemma 6.** *Let $\mathcal{A}$ be any probabilistic, polynomial-time adversary, and let $\langle \mathsf{D}, PK_{\mathcal{A}}, MSK_{\mathcal{A}}, \mathcal{T}\rangle$ be the output resulting from the adversary playing the traceability attack game $\boldsymbol{G}^{m}_{\mathsf{trt}}(1^k)$ with the challenger. Also let $\mathcal{T} \doteq \{t_1, \ldots, t_m\}$ and, for $j = 1, \ldots, m$, denote with $\boldsymbol{\delta}_{t_j}$ the compact representation of the secret key of user $t_j$ w.r.t. the public key $PK_{\mathcal{A}}$. If the pirate decoder $\mathsf{D}$ output by $\mathcal{A}$ contains a valid representation $\boldsymbol{\delta}$ for the public key $PK_{\mathcal{A}}$, such that $\boldsymbol{\delta'}$ is not a linear combination of $\boldsymbol{\delta'}_{t_1}, \ldots, \boldsymbol{\delta'}_{t_m}$, then the discrete-log problem over $\mathcal{G}$ is solvable.*

**Non-Black-Box Tracing Algorithm** We present a deterministic tracing algorithm that recovers, under Assumptions 2 and 3, the identities of the traitors that created the pirate key. Suppose that the content of a pirate decoder is exposed. By Assumption 3, it is possible to extract from $\mathsf{D}$ a valid representation $\boldsymbol{\delta}$ of the current public key $PK_{\mathcal{A}}$. Define $\{x_1, \ldots, x_n\}$ to be the set of all values assigned to the users in the system (where $n$ denotes the total number of users in the system), and let $\boldsymbol{\delta_1}, \ldots, \boldsymbol{\delta_n}$ be the corresponding secret keys. Let $\{z_{i_1}, \ldots, z_{i_v}\}$ be the set of values of the revoked users specified in the current public key.[7] Remember that the user-key of user $j$ w.r.t. the current public key can be compactly represented in the form

$$\boldsymbol{\delta_j} \doteq \langle \lambda_0^{(j)} A(x_j), \lambda_0^{(j)} B(x_j), \lambda_{i_1}^{(j)}, \ldots, \lambda_{i_v}^{(j)} \rangle$$

where $\lambda_j^{(j)}, \lambda_{i_1}^{(j)}, \ldots, \lambda_{i_v}^{(j)}$ are the Lagrange coefficients defined in Equations (3) and (4). Notice that, for any polynomial $P \in \mathbb{Z}_q^v[x]$, it holds that

$$P(0) = \lambda_0^{(j)} P(x_j) + \lambda_{i_1}^{(j)} P(x_{i_1}) + \ldots + \lambda_{i_v}^{(j)} P(x_{i_v}).$$

Consider the matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times v}$ whose $j$th row is $\boldsymbol{\delta'_j}$, $j = 1, \ldots, n$, i.e.:

$$\mathbf{A} \doteq \begin{pmatrix} \lambda_{i_1}^{(1)} \ldots \lambda_{i_v}^{(1)} \\ \ldots \\ \lambda_{i_1}^{(n)} \ldots \lambda_{i_v}^{(n)} \end{pmatrix}$$

Define the identities of the traitors to be $\{t_1, \ldots, t_m\} \subseteq \{1, \ldots, n\}$. By Lemma 6 and Assumption 2, $\boldsymbol{\delta'}$ must be a linear combination of the vectors $\boldsymbol{\delta'_{t_1}}, \ldots, \boldsymbol{\delta'_{t_m}}$ obtained by projecting the traitors' user-keys $\boldsymbol{\delta_{t_1}}, \ldots, \boldsymbol{\delta_{t_m}}$ onto the last $v$ components. It follows that $\boldsymbol{\delta'}$ also lies in the linear span of $\boldsymbol{\delta'_1}, \ldots, \boldsymbol{\delta'_n}$. More precisely, there exists a vector $\boldsymbol{\varphi}$ of Hamming weight at most $m$ such that

$$\boldsymbol{\delta'} = \boldsymbol{\varphi} \cdot \mathbf{A}. \tag{35}$$

Consider the two matrices:

$$\mathbf{B} \doteq \begin{pmatrix} x_{i_1} \ldots x_{i_1}^v \\ \ldots \\ x_{i_v} \ldots x_{i_v}^v \end{pmatrix} \quad \mathbf{H} \doteq \begin{pmatrix} -\lambda_0^{(1)} x_1 \ldots & -\lambda_1^1 x_1^v \\ \ldots \\ -\lambda_0^{(n)} x_n \ldots & -\lambda_0^{(n)} x_n^v \end{pmatrix}$$

It is easy to verify that $\mathbf{A} \cdot \mathbf{B} = \mathbf{H}$. Multiplying (35) by $\mathbf{B}$, we get

$$\boldsymbol{\varphi} \cdot \mathbf{H} = \boldsymbol{\delta''}$$

[7] W.l.o.g. we are assuming that the current saturation level $L$ is equal to $v$.

where

$$\boldsymbol{\delta''} \doteq \boldsymbol{\delta'} \cdot \mathbf{B}. \tag{36}$$

Let $\mathcal{C}$ denote the linear code over $\mathbb{Z}_q^n$ that has $\mathbf{H}$ as its parity-check matrix, i.e.

$$\boldsymbol{c} \in \mathcal{C} \Longleftrightarrow \boldsymbol{c} \cdot \mathbf{H} = \mathbf{0}.$$

Let $\lambda_1, \ldots, \lambda_n$ be the Lagrange coefficients corresponding to $\{x_1, \ldots, x_n\}$; thus, for all $P \in \mathbb{Z}_q^{<n}[x]$, it holds that

$$P(0) = \lambda_1 P(x_1) + \ldots + \lambda_n P(x_n).$$

In Lemma 7 (Section 6.3), we prove that $\mathcal{C}$ is a Generalized Reed-Solomon Code (GRS), with distance $(v+1)$. For more details about Generalized Reed-Solomon Codes, see e.g. [16]. Generalized Reed-Solomon Codes can be decoded efficiently by the algorithm of Berlekamp and Welch [1]. This means that, for any $e \leq m$ and any vector $\boldsymbol{\mu} \in \mathbb{Z}_q^n$, there exists (at most) a unique vector $\boldsymbol{\omega} \in \mathcal{C}$ that disagrees with $\boldsymbol{\mu}$ in at most $e$ positions (since $\mathcal{C}$ has distance $(v+1)$ and $m = \lfloor \frac{v}{2} \rfloor$). Moreover, such unique vector $\boldsymbol{\omega} \in \mathcal{C}$ (if it exists) can be recovered in deterministic polynomial-time. We now describe how this can be exploited to reconstruct $\boldsymbol{\varphi}$ given $\boldsymbol{\delta'}$.

First, we compute an arbitrary vector $\boldsymbol{\vartheta} \in \mathbb{Z}_q^n$ that satisfies the system of equations

$$\boldsymbol{\vartheta} \cdot \mathbf{H} = \boldsymbol{\delta''}. \tag{37}$$

where $\boldsymbol{\delta''}$ is defined in Equation (36). Note that such $\boldsymbol{\vartheta}$ can be found by standard linear algebra since Equation (37) induces a system of $v$ equations with $n$ unknowns, $n > v$, and $\mathbf{H}$ contains a non-singular minor of size $v$. It is easy to verify that the vector

$$\boldsymbol{\omega} \doteq \boldsymbol{\vartheta} - \boldsymbol{\varphi}$$

belongs to the linear code $\mathcal{C}$; indeed,

$$\begin{aligned} \boldsymbol{\omega} \cdot \mathbf{H} &= \boldsymbol{\vartheta} \cdot \mathbf{H} - \boldsymbol{\varphi} \cdot \mathbf{H} \\ &= \boldsymbol{\delta''} - \boldsymbol{\delta''} \\ &= \mathbf{0}. \end{aligned}$$

As a result, the vector $\boldsymbol{\vartheta}$ can be expressed as $\boldsymbol{\vartheta} = \boldsymbol{\omega} + \boldsymbol{\varphi}$.

Provided that the number of traitors is at most $m$, it holds that the Hamming weight of $\boldsymbol{\varphi}$ is less than or equal to $m$ and as a result $\boldsymbol{\vartheta}$ is an $n$-vector that differs in at most $m$ positions from the vector $\boldsymbol{\omega}$ (which belongs to $\mathcal{C}$): in other words, we can view $\boldsymbol{\vartheta}$ as a "partially corrupted" version of the codeword $\boldsymbol{\omega}$. Therefore, we can recover $\boldsymbol{\omega}$ from $\boldsymbol{\vartheta}$, by running the Berlekamp-Welch decoding algorithm for GRS-codes on input $\boldsymbol{\vartheta}$. At this point, $\boldsymbol{\varphi}$ can be computed as $\boldsymbol{\varphi} = \boldsymbol{\vartheta} - \boldsymbol{\omega}$.

By Equation (35), $\boldsymbol{\varphi}$ is a vector of Hamming weight at most $m$, whose non-zero components correspond to the identities of the traitors; thus, the traitors' identities can be recovered as

$$\{t_1, \ldots, t_m\} = j \in \{1, \ldots, n\} \wedge \varphi_j \neq 0\}.$$

Time-Complexity. The tracing procedure has time complexity $\mathcal{O}(n^2)$, which can be optimized 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 $m$, it is still possible to extract candidate sets of potential traitors using the Guruswami-Sudan algorithm [13], which performs GRS-decoding "beyond the error-correction bound". This will work provided that the size of the traitor coalition is less than or equal to $n - \sqrt{n(n-v)}$.

**Correctness of Non-Black-Box Tracing** Given Lemmas 6 and 7, the correctness of the non-black-box tracing algorithm described above follows from the properties of algebraic decoding of GRS codes. Thus, to conclude the argument, we now move on to the proofs of these Lemmas.

**Proof of Lemma 6**
Let $g$ be a generator of $\mathcal{G}$, and let $g' \doteq g^w$. Using adversary $\mathcal{A}$ described in the attack game $\mathbf{G}^m_{\mathsf{trt}}(1^k)$, we want to show how to recover the value $w$. In performing step 1. of $\mathbf{G}^m_{\mathsf{trt}}(1^k)$, choose two random polynomials $A^0(x)$ and $B^0(x)$ and set the initial public key to be

$$\langle g, g', g^{A^0(0)} g'^{B^0(0)}, \langle \ell, g^{A^0(\ell)} g'^{B^0(\ell)} \rangle_{\ell=1}^{v} \rangle.$$

The game then proceeds as described in Section 6.1; in particular, let $\bar{t}$ be the number of $\mathsf{New\text{-}period}$ operation occuring during the entire game. Eventually, adversary $\mathcal{A}$ outputs a pirate decoder $\mathsf{D}$ from which (by Assumption 3) it is possible to extract a vector

$$\boldsymbol{\delta} = \langle \gamma_a, \gamma_b, \gamma_1, \dots, \gamma_v \rangle,$$

which is a valid representation of the final public key $PK_{\mathcal{A}}$. In formula,

$$y = g^{\gamma_a} g'^{\gamma_b} \prod_{\ell=1}^{v} h_\ell^{\gamma_\ell} \tag{38}$$

where

$$PK_{\mathcal{A}} \doteq \langle g, g', y, \langle x_{i_\ell}, h_\ell \rangle_{\ell=1}^{v} \rangle.$$

Considering discrete logarithms to the base $g$ of Equation (38), we get:

$$A^{\bar{t}}(0) + wB^{\bar{t}}(0) = \gamma_a + \sum_{\ell=1}^{v} A^{\bar{t}}(x_{i_\ell})\gamma_\ell + w\Big(\gamma_b + \sum_{\ell=1}^{v} B^{\bar{t}}(x_{i_\ell})\gamma_\ell\Big)$$

that can be rewritten as:

$$w\Big(\gamma_b + \sum_{\ell=1}^{v} B^{\bar{t}}(x_{i_\ell})\gamma_\ell - B^{\bar{t}}(0)\Big) = A^{\bar{t}}(0) - \gamma_a - \sum_{\ell=1}^{v} A^{\bar{t}}(x_{i_\ell})\gamma_\ell \tag{39}$$

Notice that both the right-hand side and the coefficient of $w$ in (39) are known, so that if such coefficient is non-zero (or, equivalently, if the right-hand side of (39) is non-zero), then we can successfully recover the value of $w$, thus violating Assumption 2. To complete the argument, it then suffices to show that the right-hand side of (39) is zero only with negligible probability, or equivalently that:

$$\Pr[\gamma_a = \bar{\gamma}_a] = 1/q \tag{40}$$

where

$$\bar{\gamma}_a \doteq A^{\bar{t}}(0) - \sum_{\ell=1}^{v} A^{\bar{t}}(x_{i_\ell}).$$

To this aim, below we prove that, conditioning on all the other information in $\mathcal{A}$'s view, the quantity $\bar{\gamma}_a$ is uniformly distributed in $\mathbb{Z}_q$. It will follow that $\mathcal{A}$'s chances of outputting a value $\gamma_a$ equal to $\bar{\gamma}_a$ are just 1 in $q$, proving Equation (39) and thus the Lemma.

To prove that $\bar{\gamma}_a$ is distributed uniformly in $\mathbb{Z}_q$, we again make use of Lemma 1 following the same approach described in Section 5.2.

Consider the quantity

$$\boldsymbol{V} \doteq (\mathsf{Coins}, w, \{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}})$$

where $\mathsf{Coins}$ represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, and $\{\{c_j^t, r_j^t\}_{j=1}^{2v+2}\}_{t=1}^{\bar{t}}$ represents all the randomness used in the $\bar{t}$ $\mathsf{New\text{-}period}$ operations that took place during the $\mathbf{G}_{\mathsf{trt}}^m(1^k)$ attack game.

The remaining randomness used during the attack game consists of the $2v+2$ coefficients of the polynomials $A^0(\cdot)$, $B^0(\cdot)$ and can be represented by a vector $\boldsymbol{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$:

$$\boldsymbol{\alpha} \doteq (a_0, a_1, \ldots, a_v, b_0, b_1, \ldots, b_v)^T.$$

Consider the vector $\boldsymbol{\beta} \in \mathbb{Z}_q^{(v+m+2)\times 1}$ defined as:

$$\boldsymbol{\beta} \doteq (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_v, \mathbf{A}_1, \ldots, \mathbf{A}_m, \bar{\gamma}_a)^T$$

where $\mathbf{X}_0 \doteq A^0(0) + wB^0(0)$, $\mathbf{X}_\ell \doteq A^0(\ell) + wB^0(\ell)$, for $\ell = 1.\ldots, v$ and $\mathbf{A}_j \doteq A^0(t_j)$ for $j = 1, \ldots, m$.

It is clear by inspection that all the information in the view of the adversary $\mathcal{A}$ during the attack game $\mathbf{G}_{\mathsf{trt}}^m(1^k)$ is completely determined by $\boldsymbol{V}$ and $\boldsymbol{\beta}$. In particular, the initial public key $PK^0$ is fixed by $\boldsymbol{\beta}$ and $w$, and the secret keys of the traitors are determined by the choice of $\boldsymbol{\beta}$, $\mathsf{Coins}$ and $w$.

The quantities in $\boldsymbol{V}$, $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are related according to the following matrix equation:

$$\boldsymbol{\beta} = \mathbf{M} \cdot \boldsymbol{\alpha} + \boldsymbol{\gamma}$$

where $\boldsymbol{\gamma} \in \mathbb{Z}_q^{(v+m+2)\times 1}$ is the vector

$$\boldsymbol{\gamma} \doteq \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ D^{0,\bar{t}}(0) - \sum_{\ell=1}^{v} D^{0,\bar{t}}(x_{i_\ell})\gamma_\ell \end{pmatrix}$$

and $\mathbf{M} \in \mathbb{Z}_q^{(v+m+2)\times(2v+2)}$ is the matrix

$$\begin{pmatrix} 1 & 0 & \ldots & 0 & w & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 1 & w & w & \ldots & w \\ & & \vdots & & & & \vdots & \\ 1 & v & \ldots & v^v & w & wv & \ldots & wv^v \\ 1 & x_{t_1} & \ldots & x_{t_1}^v & 0 & 0 & \ldots & 0 \\ & & \vdots & & & & \vdots & \\ 1 & x_{t_m} & \ldots & x_{t_m}^v & 0 & 0 & \ldots & 0 \\ 1 - \sum_{\ell=1}^{v} \gamma_\ell & -\sum_{\ell=1}^{v} \gamma_\ell x_{i_\ell} & \ldots & -\sum_{\ell=1}^{v} \gamma_\ell x_{i_\ell}^v & 0 & 0 & \ldots & 0 \end{pmatrix}$$

By inspection, it is possible to see that the first $v+m+1$ rows of $\mathbf{M}$ are linearly independent, provided that $w \neq 0$. To see that the rank of $\mathbf{M}$ is indeed $v+m+2$, define $\mathbf{T} \in \mathbb{Z}_q^{m\times v}$ to be the minor of matrix $\mathbf{A}$ resulting from considering only rows $t_1, \ldots, t_m$:

$$\mathbf{T} \doteq \begin{pmatrix} \lambda_{i_1}^{(t_1)} & \ldots & \lambda_{i_v}^{(t_1)} \\ & \ldots & \\ \lambda_{i_1}^{(t_m)} & \ldots & \lambda_{i_v}^{(t_m)} \end{pmatrix}$$

It is possible to show that if the last row of $\mathbf{M}$ were in the linear span of the first $v+m+1$ rows of $\mathbf{M}$, it would follow that $\boldsymbol{\delta}'$ should belong to the linear span of the rows of $\mathbf{T}$. But since, by hypothesis, $\boldsymbol{\delta}'$ is not a linear combination of $\boldsymbol{\delta}'_{t_1}$, …, $\boldsymbol{\delta}'_{t_m}$, the matrix $\mathbf{M}$ must have full rank.

As soon as we fix $\boldsymbol{V}$, the first $v+m+1$ entries of $\boldsymbol{\gamma}$ and the first $v+1$ rows of $\mathbf{M}$ are determined, but $\boldsymbol{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. Similarly to the proof of Lemma 3, it is also possible to show that fixing the first $v+j+1$ entries of $\boldsymbol{\beta}$ determines the $(v+j+2)$th row of $\mathbf{M}$, for $j=1,\ldots,m$; and that moreover, fixing the first $v+m+1$ entries of $\boldsymbol{\beta}$ also determines the last rows of $\boldsymbol{\gamma}$ and of $\mathbf{M}$.

Hence, by Lemma 1, we can conclude that the conditional distribution of $\bar{\gamma}_a$ w.r.t. $\boldsymbol{V}$, and to the first $v+m+1$ entries of $\boldsymbol{\beta}$, is uniform over $\mathbb{Z}_q$. In other words, conditioning on all the other information in $\mathcal{A}$'s view, the quantity $\bar{\gamma}_a$ is uniformly distributed over $\mathbb{Z}_q$. Equation (39), and thus the Lemma, follows. □

**Lemma 7.** *Consider the Generalized Reed-Solomon code:*

$$\mathcal{C}' \doteq \{\langle -\frac{\lambda_1}{\lambda_0^{(1)}}P(x_1),\ldots,-\frac{\lambda_n}{\lambda_0^{(n)}}P(x_n)\rangle \mid P \in \mathbb{Z}_q^{<n-v}[x]\}.$$

*It holds that*

1. $\mathcal{C}=\mathcal{C}'$.
2. $\mathcal{C}$ *is a linear code with message-rate* $(n-v)/n$ *and distance* $v+1$.

*Proof.*
1. We only need to show that $\mathcal{C}' \subseteq \mathcal{C}$. Indeed, assuming that $\mathcal{C}'$ is a linear sub-space of $\mathcal{C}$, since $\dim(\mathcal{C}) = n-v = \dim(\mathcal{C}')$, it immediately follows that $\mathcal{C}=\mathcal{C}'$.

To prove that $\mathcal{C}' \subseteq \mathcal{C}$, notice that if $\langle c_1,\ldots,c_n\rangle \in \mathcal{C}'$, then it is of the form

$$\big\langle -\frac{\lambda_1}{\lambda_0^{(1)}}P(x_1),\ldots,-\frac{\lambda_n}{\lambda_0^{(n)}}P(x_n)\big\rangle$$

for some polynomial $P \in \mathbb{Z}_q^{<n-v}[x]$. We now verify that $\langle c_1,\ldots,c_n\rangle$ belongs to $\mathcal{C}$. First, notice that for $\ell = 1,\ldots,v$, multiplying $\langle c_1,\ldots,c_n\rangle$ by the $\ell$th column of $\mathbf{H}$ we get

$$\langle c_1,\ldots,c_n\rangle \cdot \langle -\lambda_0^{(1)}x_1^\ell,\ldots,-\lambda_0^{(n)}x_n^\ell\rangle = \sum_{i=1}^n \lambda_i P(x_i)x_i^\ell.$$

Now observe that

$$\sum_{i=1}^n \lambda_i P(x_i)x_i^\ell = 0$$

by the choice of $\lambda_1,\ldots,\lambda_n$ and the facts that degree$(P) < n-v$ and $\ell \le v$ (just consider the polynomial $Q(x) \doteq P(x)x^\ell \in \mathbb{Z}_q^{<n}[x]$). It follows that

$$\langle c_1,\ldots,c_n\rangle \cdot \mathbf{H} = \mathbf{0}.$$

2. Observe that a vector of $\mathbb{Z}_q^{n-v}$ can be encoded as the coefficients of a polynomial $P \in \mathbb{Z}_q^{<n-v}[x]$. The corresponding codeword of $\mathcal{C}$ will be the vector

$$\big\langle -\frac{\lambda_1}{\lambda_0^{(1)}}P(x_1),\ldots,-\frac{\lambda_n}{\lambda_0^{(n)}}P(x_n)\big\rangle.$$

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. □

## 7 Conclusions and Future Work

We introduce the first public-key traitor tracing scheme where an unlimited number of users can be efficiently added and removed from the system. Our scheme enjoys both client-side scalability, by supporting a dynamically-changing user population, and server-side scalability, as it enables many content providers to use a common content distribution infrastructure.

We present a formal model for scalable public-key traitor tracing, and a thorough analysis of the revocation and tracing properties of our scheme against adaptive adversaries.

At a technical level, our adversarial model improves over previous modeling for public-key traitor tracing by capturing a larger class of adversaries, endowed with greater control over system, than what previously considered in the related literature. In particular, in our model the adversary can control an *a priori* unbounded number of user additions and removals. The main limitation of our formal model is that the adversary is supposed to be fully revoked in a "window" of the system.

An interesting open problem left open by our research consists of extending our results to a more general adversarial model, in which the adversary is not supposed to obey the "window" constrain.

## Acknowledgments

We are grateful to Antonio Nicolosi for his constructive criticism and encouraging support throughout this research.

We thank the anonymous referees for helping in improving the readability of the paper, and one of them in particular for pointing out a flaw in an early version of our scheme.

## References

1. E.R. Berlekamp and L. R. 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. LNCS 1666. Full version available at `crypto.stanford.edu/~dabo/pubs.html`.
4. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates—Building in Privacy.* PhD thesis, Technical University of Eindhoven, 1999.
5. 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.
6. B. Chor, A. Fiat, and N. Naor. Tracing Traitors. In *Advances in Cryptology—Crypto '94*, pages 257–270. Springer-Verlag, 1994. LNCS 839.
7. R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Scheme Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
8. Y. Dodis and N. Fazio. Public-Key Broadcast Encryption for Statless Receivers. In *Digital Rights Management—DRM '02*, pages 61–80. Springer, 2002. LNCS 2696.
9. 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.
10. A. Fiat and M. Naor. Broadcast Encryption. In *Advances in Cryptology—Crypto '93*, pages 480–491. Springer-Verlag, 1993. LNCS 773.
11. 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.
12. 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.
13. 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.

14. 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.
15. 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.
16. F. J. MacWilliams and N. Sloane. *The Theory of Error Correcting Codes*. North Holland, Amsterdam, 1977.
17. 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.
18. M. Naor and B. Pinkas. Threshold Traitor Tracing. In *Advances in Cryptology—Crypto '98*, pages 502–517. Springer-Verlag, 1998. LNCS 1462.
19. M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Financial Cryptography—FC 2000*, pages 1–20. Springer-Verlag, 2000. LNCS 1962. Full version available at `www.wisdom.weizmann.ac.il/~naor/onpub.html`.
20. 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.
21. 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.
22. D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. Available at `ftp://ftp.ietf.org/rfc/rfc2627.txt`, 1997.