On Cryptographic Techniques for Digital Rights Management

by

Nelly Fazio

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Courant Institute of Mathematical Sciences

New York University

September 2006

____

Yevgeniy Dodis

To Antonio

Acknowledgments

This thesis literally would not have been possible without the impeccable supervision of my advisor, Yevgeniy Dodis. During these years at New York University, Yevgeniy provided invaluable guidance through the technical hurdles of my dissertational work, and regular interaction with him helped shaping my approach to research in general. Yevgeniy believed in me from the beginning, and his encouragement greatly helped me maintaining a high motivation through the inevitable ups and downs of the Ph.D. studies. For all of this, I am deeply thankful to him.

Most of the material in this thesis is the result of joint work with Yevgeniy Dodis, Aggelos Kiayias, Anna Lysyanskaya, Antonio Nicolosi, Duong Hieu Phan, Danfeng Yao and Moti Yung. I would like to express my gratitude for their contributions to all of them. Many thanks also to my other collaborators: Mike Atallah, Marina Blanton, Emmanuel Bresson, Dario Catalano, Ivan Damgård and Keith Frikken. Working with you was at the same time fruitful and enjoyable. Thanks to you all.

Sincere thanks go also to Victor Shoup and to my fellow graduate students in the NYU Crypto Reading Group: Siddhartha Annapureddy, Carl Bosley, Sze Ming (Sherman) Chow, Michael Freedman, Kristiyan Haralambiev, Antonio Nicolosi, Prashant Puniya, Roberto Oliveira and Shabsi Walfish. Our weekly meetings have played a central role in widening my perspective on cryptographic research: many thanks to Yevgeniy for starting this seminar series and to all internal and external speakers for keeping it alive!

My summer internship at Aarhus University, Denmark, will always be a very dear memory to me. In particular, the friendliness with which I was hosted by the members of the local Crypto Reading Group was very much felt and appreciated. Special thanks to Ivan Danmgård, for having made my stay possible and for being such a wonderful person to work with, both professionally and humanely.

Similarly, I am very thankful to Jacques Stern and Dario Catalano for the opportunity of spending a productive summer with "l' èquipe Crypto" at Ècole Normale Supérieure in Paris, France.

I would also like to thank Dan Boneh and David Mazières and all the members of the Security Lab at Stanford University for hosting me during the Fall of 2005.

I am very thankful to Michael Overton for his deligent supervision of my progress in the Ph.D. program, and for his caring attention to my social and academic integration within the Computer Science Department at NYU.

My experience at NYU would certainly have been less enjoyable without the fun time spent with my officemates Kranthi Gade, Alyssa Lees, Emre Mengi and Ziyang Wang. I am especially thankful to Emre for his sincere friendship and for always being ready to lend a hand.

My deepest gratitude goes to my family back in Italy, and in particular to my parents, Orazio and Ernestina, and to my brother Giuseppe. Even from across the ocean the warmth of their support was always felt strong, and their encouragement constant and indispensable.

All my love goes to my fiancé Antonio Nicolosi, who is also my personal and professional guide: I especially would like to thank him for all the love, joy and support with which he constantly overwhelms me!

Abstract

With more and more content being produced, distributed, and ultimately rendered and consumed in digital form, devising effective Content Protection mechanisms and building satisfactory Digital Rights Management (DRM) systems have become top priorities for the Publishing and Entertaining Industries.

To help tackle this challenge, several cryptographic primitives and constructions have been proposed, including mechanisms to securely distribute data over a unidirectional insecure channel (Broadcast Encryption), schemes in which leakage of cryptographic keys can be traced back to the leaker (Traitor Tracing), and techniques to combine revocation and tracing capabilities (Trace-and-Revoke schemes).

In this thesis, we present several original constructions of the above primitives, which improve upon existing DRM-enabling cryptographic primitives along the following two directions: (1) Widening their scope of applicability *e.g.*, by considering models taking into accounts usability issues typical of the DRM setting; and (2) Strengthening their security guarantees to higher levels that are standards, for example, in the case of stand-alone encryption.

Our results along the first line of work include the following:

- An efficient public-key broadcast encryption scheme, which allows mutually mistrusting content providers to leverage a common delivery infrastructure, and can cope with low-end, stateless receivers;

- A traitor tracing scheme with optimal transmission rate, in which encryption does not cause a blow-up in the size of the content, thus allowing for optimal utilization

of the broadcast channel;

- A public-key tracing and revoking scheme that can deal with both server-side and client-side scalability issues, while preserving traceability.

As for the second direction, our contribution can be divided as follows:

- A forward-secure public-key broadcast encryption scheme, in which the unauthorized access resulting from cracking a user-key is constrained to a minimal time frame which is delimited, in the future, by the revocation mechanism, and in the past, by forward secrecy;
- A precise formalization of the notion of adaptive chosen-ciphertext security for public-key broadcast encryption schemes, along with a modular and efficient construction.

Overall, the cryptographic tools developed in this thesis provide more flexibility and more security than existing solutions, and thus offer a better match for the challenges of the DRM setting.

Contents

List of Figures

Chapter 1

Introduction

We live in the digital information age: the growth of new technologies for the communication infrastructure and the wide availability of digital storage devices have resulted in most content being produced, distributed, and ultimately rendered and consumed by end users in a digital form.

The intrinsic nature of digital data makes the task of manipulating, replicating and (re-)distributing digital copies extremely easy. Publishing content covered by intellectual property or copyright is then faced by the challenge of countering illegal access to the digital good, while still enabling customers to exercise their usage rights (*e.g.* rendering, processing, backup, etc.) on purchased content.

Digital Rights Management (DRM) consists of architectures, protocols and technologies aimed at providing satisfactory solutions to the above problem. The complexity of the problem is apparent from the outset, and indeed DRM has solicited efforts from a variety of research areas, spreading from law and ethics to economics, from logics and formal languages to system architecture.

Managing rights on digital content involves describing, monitoring and tracking all forms of usage of the copyrighted material. Describing legitimate usages with suitable policy languages plays an important role in guaranteeing that digital goods are distributed in compliance with legislation safeguarding the rights of the customers; at the same time, it provides guidelines that enable monitoring mechanisms to discern legal usage from abuse of the digital content.

Keeping track of how content is being used and who is using it provides an effective (and often the only) way to trace episodes of abuse back to the misbehaving user(s). However, in pursuing these goals, an actual DRM system ought to deal carefully with privacy concerns—tracing all legal usages by honest customers is clearly an unacceptable measure for any society that values individual liberty. Such tension between technical issues and social concerns adds to the complexity of the problem, but a suitable balance is a necessary requirements for the development of a successful DRM system: A design inspired only by the technical need to prevent piracy, but irrespective of the customers' legitimate expectations about the modes of fruition of the purchased goods, may well succeed at achieving zero-piracy, but it would likely still fail due to the customers base indifference (or even resistance) toward its adoption.

In all cases, however, at the core of any DRM proposal there always ought to be a mechanism to enforce compliance to the prescribed rules and polices, which naturally calls for the employment of cryptographic techniques.

The focus of this doctoral investigation is on the Cryptography that is needed to enable DRM technologies. Several cryptographic primitives and constructions have been proposed in the last decade to tackle this challenge, including mechanisms to securely distribute data over a unidirectional insecure channel (Broadcast Encryption), schemes in which leakage of cryptographic keys can be traced back to the leaker (Traitor Tracing), and techniques to combine revocation and tracing capabilities (Trace-and-Revoke schemes).

Some of these constructions, however, suffer either from inadequacy in the level of security they guarantee, or from limitedness of the underling model with respect to the complexity of the specific aspect of the DRM challenge that they aim to address.

In this thesis, we seek to ameliorate such limitations in DRM-enabling cryptographic primitives by investigating novel constructions for existing primitives that achieve higher security standards, and by developing new models, capturing (some of the) practical issues that arise in the engineering of a wider range of DRM applications.

1.1 Our Contributions

Corporate DRM efforts have so far been inspired by the underlying belief that existing cryptographic tools, "*as is*," are sufficient to build satisfactory DRM systems, and that DRM solutions could be attained by engineering the system so as to maintain a rigid control over the medium used by the consumers to render the digital content. However, such approach has not yet proven successful; rather, the failure upon which some of the major companies in this industry have stumbled recently has been spectacular (*e.g.* see [5]).

It is interesting to notice that such failed solutions were not based on any of the cryptographic primitives that have been proposed in the last decade to help tackle the DRM problem. Such neglect may have been due just to ignorance of the relevant cryptographic literature. However, it seems unlikely that the proposed crypto-DRM constructions could have been employed anyway, for we believe that, in their current form, they are not applicable to complex usage scenarios, and offer only limited security guarantees. To address these shortcomings, new tools need to be developed to better match the specific challenges of the DRM setting. Toward this end, this thesis presents several original constructions, which improve upon existing DRM-enabling cryptographic primitives along the following two directions: (1) widening their scope of applicability *e.g.*, by considering models taking into accounts usability issues typical to the DRM setting; and (2) strengthening their security guarantees to higher levels that are standards, for example, in the case of stand-alone encryption.

1.2 Widening the Scope of Applicability

1.2.1 Public-Key Broadcast Encryption for Stateless Receivers

A *broadcast encryption* scheme (*cf.* Chapter 3, Section 3.1) allows data to be securely distributed to a dynamically changing set of users over an insecure channel. The relevance of broadcast encryption for DRM technologies stems from its inherent access-control capability, which could be leveraged *e.g.*, for pay-TV systems, distribution of copyrighted material or streaming audio/video.

In symmetric-key broadcast encryption schemes, only the trusted designer of the system can broadcast data to the receivers, because encrypting content requires knowledge of sensitive information whose disclosure would compromise the security of the entire scheme. In public-key schemes, in contrast, the trusted designer of the system publishes a short public key which enables anybody to broadcast data. This allows mutually mistrusting content providers to share a common broadcast medium to securely disseminate information to their own user population, thus minimizing the overhead associated with the maintenance of the broadcasting infrastructure. Moreover, each user will need to store only one piece of secret information, thus reducing the storage requirement at the customer side.

The wider applicability of the public-key scenario makes it a better candidate for practical use, but symmetric-key solutions have traditionally received much more attention in the cryptographic literature. As a result, the state-of-the-art for this setting has usually been more advanced than for the public-key model. As first contribution to the field of DRM-enabling cryptographic techniques, in Chapter 4 we describe how to bridge the gap between the efficiency of the state-of-the-art broadcast encryption solution for the symmetric-key *versus* public-key setting, thus combining the "best of both worlds."

The state-of-the art symmetric-key broadcast encryption scheme was proposed in [65], where the authors presented the *Subset-Cover Framework* as a formal environment within which to formally define and analyze the security of revocation schemes. As specific examples, the *Complete Subtree* (CS) method and the *Subset Difference* (SD) method were formalized and proven secure within this framework. (Subsequently, in [51] the *Layered Subset Difference* (LSD) method was introduced as an improvement on the SD method: our techniques also apply to this improved scheme. See Chapter 4, Section 3.3 for more details.)

The work of [65] also briefly considered the question of transposing any Subset-Cover revocation scheme to the asymmetric setting, mentioning a generic, but highly inefficient, technique for this purpose. Naor *et al.* hinted that for the case of their basic scheme (the CS method), tools from Identity-Based Cryptography were likely to be sufficient to regain the lost efficiency. However, a solution for the case of the more advanced SD method was left as an interesting open problem.

In Chapter 4, we solve this problem, showing that any Hierarchical Identity-Based Encryption (HIBE) scheme can be used to obtain a public-key counterpart for the SD method that matches the efficiency parameters of the original symmetric-key scheme. *En route*, we also verify that regular Identity-based Encryption can indeed support a public-key version of the CS method. We also show that our HIBE-based technique can be used for the LSD variant as well.

1.2.2 Traitor Tracing with Optimal Transmission Rate

Traitor tracing schemes (*cf.* Chapter 3, Section 3.2) are multi-recipient encryption algorithms where the secret key of each user is fingerprinted, so that it can be traced back in case of leakage. In Chapter 6, we describe the design of the first traitor tracing scheme with efficient black-box traitor tracing in which the ratio of the ciphertext and plaintext lengths (the *transmission rate*) is asymptotically 1, which is optimal. Optimal transmission rate is of major importance for concrete DRM systems (*e.g.* for Pay-per-View systems transmitting live sport events), since it entails an optimal utilization of the communication medium used to distribute the encrypted content. Previous constructions in this setting either obtained constant (but not optimal) transmission rate [57], or did not support black-box tracing [27].

As additional contributions, we point out and resolve an issue in the black-box traitor tracing mechanism in the scheme of [57]; and we show that the scheme of [27], which extends [57] and inherits its tracing mechanism, in fact does not provide black-box tracing nor (local) public traceability. In particular, whereas fixing the scheme of [57] requires just a simple, local change to the tracing algorithm, repairing the black-box functionality and the public traceability features of [27] voids the claimed optimality of the transmission rate, and results in a new scheme with essentially the same parameters as in [57].

Our construction is based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model, and attains the same features of public traceability as (a repaired variant of) [27], which instead is less efficient and requires non-standard assumptions for bilinear groups.

1.2.3 Scalable Public-Key Tracing and Revoking

Traitor tracing schemes deter subscribers of a content distribution system from leaking their keys by the mere fact that the identities of the leakers (the *traitors*) can be uncovered. However, in case of leakage of a decryption key, a tracing scheme does not provide by itself a way to exclude from subsequent broadcasts the misbehaving users that have been identified as traitors. For this reason, traitor tracing solutions are most useful when combined with broadcast encryption techniques (which provide revocation capabilities), resulting in the so-called *trace-and-revoke* approach (*cf.* Chapter 3, Section 3.3).

In Chapter 7, we describe an extension to the trace-and-revoke approach to deal with practical issues inherent to the deployment of real systems, yielding a new notion that we call *scalable* trace-and-revoke schemes. Scalability is particularly relevant to DRM, as large scale could arguably help sustain the economical profitability of DRM services in the long run. In the context of content distribution, scalability has two facets: *server-side* and *client-side*.

Server-side scalability deals with the property that allows the population of content providers to change dynamically. Each content provider in this setting needs access to the encryption mechanism, so that it can scramble content. This suggests that each content provider needs to have the encryption keys that allow *all users* of the system to get its content. If the content providers are few and closely connected to the security manager, then one may assume that the encryption/decryption keys are shared among the providers and the security manager. But this scenario does not scale, since the likelihood of corruption of one provider (which would immediately compromise the security of the entire system) increases with the number of content providers. This leads to the need of employing public-key cryptography for server-side scalability.

Client-side scalability deals with the fact that we have a user population that is changing dynamically due to the service subscription model and security constraints. To allow for a scalable management of user accounts, keys should be easy to generate and revoke. An adversarial coalition that controls some of the user keys that have been revoked should be incapable of recovering the content. It is also important to have a mechanisms to identify misbehaving users to allow the piracy-deterrence property while the population is dynamically changing.

Based on these design guidelines, in Chapter 7, we presented the first model of a scalable public-key traitor tracing scheme where an unlimited number of users can be added and removed efficiently from the system. We present a concrete scheme meeting these requirements, based on the Decisional Diffie-Hellman DDH assumption. Addition of users does not affect the keys of 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*; within a period, a bounded number of user removals can be executed; unlimited number of user removals is then achieved by the implementation of an *efficient* New-period operation (where "efficient" here means that the operation does not depend on the total number of users and that it does not require private channels between the system manager and the users). Within a period, users are not required to maintain state. With every New-period operation, though, each user needs to update its private-key information by employing an efficient key-update operation.

None of the existing proposals provided such scalability properties, while preserving traceability.

1.3 Strengthening the Security Guarantees

1.3.1 Forward-Secure Public-Key Broadcast Encryption

An important line of investigation that we pursue in this thesis is the application of the paradigm of *forward secrecy* to the setting of public-key broadcast encryption. Initially proposed in the context of digital signatures, forward secrecy aims at guaranteeing that the security properties of past applications of the private key are retained, even in the event that such key will eventually be compromised.

Apart from being interesting in its own right, forward secrecy is especially needed for broadcast encryption, where by design any party can freely listen to and store any broadcast. Should a hacker ever succeed in recovering *any* user's private key, she will manage to decrypt all past broadcast that such user was authorized to receive, *unless* the scheme enjoys forward secrecy. In other words, adding forward secrecy enables the system to constrain unauthorized access to the broadcast material to a *minimal* time frame which is delimited, in the future, by the security of the revocation mechanism, and in the past, by forward secrecy.

In Chapter 5, we construct the first forward-secure public-key broadcast encryption (FSBE) scheme. Our scheme resists a very strong type of *chosen-ciphertext* and *key corruption* attack, where the adversary is allowed to corrupt users (thus obtaining their secret keys) in any order, and can also ask non-corrupted users to decrypt any ciphertexts of her choice, during any time period. The security of our scheme is based on the Decisional Bilinear Diffie-Hellman Inversion (DBDHI) assumption in the standard model.

Of independent interest, as a technical tool toward achieving forward-secure broadcast encryption, in Chapter 2 we define and construct an extension of the notion of Identity-Based Encryption to complex hierarchical structures, that we term *Paired Hierarchical Identity-Based Encryption* (PHIBE). In the setting of (regular) Hierarchical Identity-Based Encryption (HIBE) [52, 48, 16], each user is associated with a single hierarchical identifier: encrypting to a given user means that only the ancestors of the user in the hierarchy can decrypt the message. In PHIBE, each user belongs to two hierarchies, and encrypting to a given user means that only a common ancestor of this user in the two hierarchies can decrypt the message. Once again, our PHIBE scheme is based on the DBDHI assumption.

In [82], we also show how to extend PHIBE to multiple hierarchies.

1.3.2 Chosen-Ciphertext Security for Trace-and-Revoke Schemes

One of the most exciting accomplishments of modern Cryptography has been the formalization of the "right" notion of security for several cryptographic tasks. It took more than a decade before the notion of *adaptive chosen-ciphertext* security emerged as the accepted security standard for encryption schemes, and several more years until an efficient construction of an encryption scheme meeting such a stringent notion of security was designed.

In the context of trace-and-revoke schemes, which involve multiple recipients of the encrypted content, and operate in open environments (and are thus inherently more vulnerable to attacks), it would seem natural to demand at least a similarly stringent security level. In Chapter 8, we introduce a precise formalization of an appropriate notion of adaptive security for public-key broadcast encryption schemes, for both chosen-plaintext (IND-ID-CPA) and chosen-ciphertext (IND-ID-CCA) attack scenarios, and propose constructions IND-ID-CPA- and IND-ID-CCA-secure under the Decisional Diffie-Hellman (DDH) assumption, with no random oracles. Our public key scheme is based on the regular Cramer-Shoup encryption scheme [31, 32], but our extension is non-trivial, as we have to resolve some difficulties inherent to the broadcast encryption setting. Our IND-ID-CCA-secure scheme has constant storage complexity and public key size proportional to the revocation threshold r. The communication complexity, as well as the encryption and decryption times are all linear in r.

As a preliminary step in our construction, we show how to modify the IND-ID-CPA-scheme of [80] to achieve a much more appropriate notion of adaptive security, while maintaining essentially the same efficiency in all the parameters (up to a factor of 2). We also provide another scheme achieving a slightly weaker (but still very strong) notion of *generalized* chosen-ciphertext security (IND-ID-gCCA) [75, 2]. As argued in [2], IND-ID-gCCA security is much more robust to syntactic changes, while still sufficient for all known uses of IND-ID-CCA security. Interestingly, all the examples separating IND-ID-CCA- and IND-ID-gCCA-secure encryption were "artificial" in a sense that they made a more complicated scheme from an already existing IND-ID-CCA-secure encryption. Our work shows the first "natural" separation, but for the setting of *broadcast* public-key encryption.

Chapter 2

Preliminaries

2.1 Some Algebraic Tools

2.1.1 Lagrange Interpolation in the Exponent

Let q be a prime and $f(x)$ a polynomial of degree z over $\mathbb{Z}_q$; let $j_0, \ldots, j_z$ be distinct elements of $\mathbb{Z}_q$, and let $f_0 = f(j_0), \ldots, f_z = f(j_z)$. Using the Lagrange Interpolation, we can express the polynomial as

$$f(x) = \sum_{t=0}^{z} (f_t \cdot \lambda_t(x))$$

where

$$\lambda_t(x) = \prod_{0 \le i \ne t \le z} \frac{j_i - x}{j_i - j_t}, \quad t = 0, \ldots, z.$$

We can now define the Lagrange Interpolation Operator as follows:

$$\mathsf{LI}(j_0, \ldots, j_z; f_0, \ldots, f_z)(x) \doteq \sum_{t=0}^{z} (f_t \cdot \lambda_t(x)).$$

Consider any cyclic group $\mathbb{G}$ of order q and a generator g of $\mathbb{G}$. For any distinct values $j_0, \ldots, j_z$ of $\mathbb{Z}_q$ and (non necessarily distinct) elements $v_0, \ldots, v_z$ of $\mathbb{G}$, let us define the Lagrange Interpolation Operator in the Exponent as:

$$\mathsf{EXP\text{-}LI}(j_0, \ldots, j_z; v_0, \ldots, v_z)(x) \doteq g^{\mathsf{LI}(j_0, \ldots, j_z; \log_g v_0, \ldots, \log_g v_z)(x)}.$$

Despite being defined in terms of discrete logarithms, the function $\mathsf{EXP\text{-}LI}$ is polynomial-time computable, since:

$$g^{\mathsf{LI}(j_0,\ldots,j_z;\log_g v_0,\ldots,\log_g v_z)(x)} = \prod_{t=0}^{z} g^{(\log_g v_t \cdot \lambda_t(x))} = \prod_{t=0}^{z} v_t^{\lambda_t(x)}.$$

We also remark on another useful property of the above operator:

$$\mathsf{EXP\text{-}LI}(j_0,\ldots,j_z; v_0^r,\ldots,v_z^r)(x) = [\mathsf{EXP\text{-}LI}(j_0,\ldots,j_z; v_0,\ldots,v_z)(x)]^r.$$

In what follows, we will refer to a function of the form $g^{f(x)}$, where $f(x)$ is a polynomial, as an EXP-polynomial.

2.1.2 Discrete Logarithm Representations

Let g be a generator of $\mathbb{G}$ and let $h_0, h_1, \ldots, h_v$ be elements of $\mathbb{G}$ 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 $\mathbb{G}$, a representation of y with respect to the base $h_0,\ldots,h_v$ is a $(v+1)$-vector

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

such that:

$$y = \prod_{\ell=1}^{v} h_\ell^{\delta_\ell}$$

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

It is well known (e.g., see [24]) that obtaining representations of a given y with respect to some base $h_0,\ldots,h_v$ is as hard as the discrete logarithm problem over $\mathbb{G}$. Furthermore, it was shown in Lemma 3.2 of [17] that if some adversary is given $m < v$ random representations of some y with respect to some base, then any additional representation that can be obtained has to be a "convex combination" of the given representations (a convex combination of the vectors $\vec{\delta_1},\ldots,\vec{\delta_m}$ is a vector $\sum_{\ell=1}^{m} \mu_\ell \vec{\delta_\ell}$ with $\sum_{\ell=1}^{m} \mu_\ell = 1$). However, the scheme presented in Chapter 7 makes use of a particular family of discrete logarithm representations, introduced below. In Section 7.6 we will see how Lemma 3.2 of [17] can be modified accordingly.

2.1.3 Leap-Vectors

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

Definition 1. *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 with respect to $P(\cdot)$ and the values $z_1, \ldots, z_v$, consists of all vectors $\vec{\alpha} \in \mathbb{Z}_q^{v+1}$ for which it holds that:*

$$P(0) = \vec{\alpha} \cdot \langle 1, P(z_1), \ldots, P(z_v)\rangle. \tag{2.1}$$

In other words, a leap-vector with respect to $P(\cdot)$ 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 $\vec{\alpha} := \langle \alpha_0, \ldots, \alpha_v\rangle$ with respect to 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 2. *Given distinct $x_i, z_1, \ldots, z_v \in \mathbb{Z}_q$, and $P(\cdot) \in \mathbb{Z}_q^v[x]$, define the leap-vector $\vec{\nu}^{x_i,P}_{z_1,\ldots,z_v}$ associated to the point $\langle x_i, P(x_i)\rangle$ with respect to $P(\cdot)$ and $z_1, \ldots, z_v$ as:*

$$\vec{\nu}^{x_i,P}_{z_1,\ldots,z_v} \doteq \langle \lambda_0^{(i)} P(x_i), \lambda_1^{(i)}, \ldots, \lambda_v^{(i)}\rangle \tag{2.2}$$

where

$$\lambda_0^{(i)} \doteq \prod_{j=1}^{v} \frac{x_i}{x_i - z_j} \tag{2.3}$$

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

$$\lambda_\ell^{(i)} \doteq \frac{z_\ell}{z_\ell - x_i} \prod_{\substack{j=1 \\ j \neq \ell}}^{v} \frac{z_\ell}{z_\ell - z_j}. \tag{2.4}$$

An important property of leap-vectors is the following:

Proposition 3. *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* $\vec{\alpha} \in \mathcal{L}^P_{z_1,\ldots,z_v}$ *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

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

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

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

where

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

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 $\vec{\alpha}$ 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 $P(0)$.

2.2 Collusion-Secure Codes

Collusion-secure codes [22, 23] provide a powerful tool against illegal redistribution of fingerprinted material in settings satisfying the following *Marking Assumption*: 1) it is possible to introduce small changes to the content at some discrete set of locations (called *marks*), while preserving the "quality" of the content being distributed; but 2) it is infeasible to apport changes to a mark without rendering the entire content "useless" *unless* one possesses two copies of the content that differ at that mark.

Below, we include a formalization of the notion of collusion-secure codes, adapted from [23].

Definition 4. *Let Σ be a finite alphabet, and $n, v \in \mathbb{Z}_{\geq 0}$. An (n, v)-code over Σ is a set of n v-tuples of symbols of Σ: $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\} \subseteq \Sigma^v$.*

Definition 5. *Let T be a subset of indices in $[1, n]$. The set of undetectable positions for T is:*

$$R_T = \{\ell \in [1, v] \mid (\forall i, j \in T).[\omega_\ell^{(i)} = \omega_\ell^{(j)}]\}.$$

Notice that for each $i \in T$, the projection of each codeword $\omega^{(i)}$ over the undetectable positions for T is the same; we denote this common projected sub-word as $\omega_{|R_T}$. By the Marking Assumption, any "useful" copy of the content created by the collusion of the users in T must result in a tuple $\bar{\omega}$ whose projection over R_T is also $\omega_{|R_T}$. This is captured by the following:

Definition 6. *The set of feasible codewords for T is:*

$$F_T = \{\bar{\omega} \in (\Sigma \cup \{?\})^v \mid \bar{\omega}_{|R_T} = \omega_{|R_T}\}.$$

Definition 7. *Let $\varepsilon > 0$ and $t \in \mathbb{Z}_{\geq 0}$. $\mathcal{C}$ is an (ε, t, n, v)-collusion-secure code over Σ if there exists a probabilistic polynomial-time algorithm $\mathcal{T}$ such that for all $T \subseteq [1, n]$ of size $\mid T \mid \leq t$, and for all $\bar{\omega} \in F_T$, it holds that:*

$$Pr[\mathcal{T}(r_{\mathcal{C}}, \bar{\omega}) \in T] \geq (1 - \varepsilon),$$

where the probability is over the random coins $r_{\mathcal{C}}$ used in the construction of the (n, v)-code $\mathcal{C}$, and over the random coins of $\mathcal{T}$.

2.3 Computational Assumptions

The decisional Diffie-Hellman (DDH) problem in $\mathbb{G}_1$:

Given (P, aP, bP, U) for some $a, b \in \mathbb{Z}_q$ and $U \in \mathbb{G}_1$, output yes if $U = abP$ and no otherwise.

Definition 8 (DDH Assumption)**.** *The DDH problem is $\varepsilon_{\mathsf{DDH}}$-hard in $\mathbb{G}_1$ if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, we have*

$$\begin{aligned} \mathsf{AdvDDH}_{\mathbb{G}_1,\mathcal{A}}(k) \doteq |&Pr[\mathcal{A}(P, aP, bP, U) = yes \mid P \xleftarrow{R} \mathbb{G}_1, a, b, \xleftarrow{R} \mathbb{Z}_q, U = abP] - \\ &Pr[\mathcal{A}(P, aP, bP, U) = yes \mid P \xleftarrow{R} \mathbb{G}_1, a, b, \xleftarrow{R} \mathbb{Z}_q, U \xleftarrow{R} \mathbb{G}_1]| < \varepsilon_{\mathsf{DDH}} \end{aligned}$$

where the probability is over the random selection of P from $\mathbb{G}_1$, of a, b from $\mathbb{Z}_q$, and over $\mathcal{A}$'s random coins.

The computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$:

Given (P, aP, bP) for random $P \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$, output $abP \in \mathbb{G}_1$.

Definition 9 (CDH Assumption)**.** *The CDH problem is $\varepsilon_{\mathsf{CDH}}$-hard in $\mathbb{G}_1$ if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, we have*

$$\mathsf{AdvCDH}_{\mathbb{G}_1,\mathcal{A}}(k) \doteq Pr[\mathcal{A}(P, aP, bP) = abP \mid P \xleftarrow{R} \mathbb{G}_1, a, b \xleftarrow{R} \mathbb{Z}_q] < \varepsilon_{\mathsf{CDH}}$$

where the probability is over the random selection of P from $\mathbb{G}_1$, of a, b from $\mathbb{Z}_q$, and over $\mathcal{A}$'s random coins.

The Discrete Logarithm (DLog) problem in $\mathbb{G}_1$:

Given (P, aP) for random $P \in \mathbb{G}_1$ and $a \in \mathbb{Z}_q$, output $a \in \mathbb{G}_1$.

Definition 10 (DLog Assumption)**.** *The* DLog *problem is* $\varepsilon_{\mathsf{DLog}}$*-hard in* $\mathbb{G}_1$ *if, for all probabilistic polynomial-time algorithms* $\mathcal{A}$*, we have*

$$\mathsf{AdvDLog}_{\mathbb{G}_1,\mathcal{A}}(k) \doteq Pr[\mathcal{A}(P, aP) = a \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a \overset{R}{\leftarrow} \mathbb{Z}_q] < \varepsilon_{\mathsf{DLog}}$$

where the probability is over the random selection of P *from* $\mathbb{G}_1$*, of* a *from* $\mathbb{Z}_q$*, and over* $\mathcal{A}$*'s random coins.*

The decisional bilinear Diffie-Hellman (DBDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$:

Given (P, aP, bP, cP, h) for random $P \in \mathbb{G}_1$, $a, b, c \in \mathbb{Z}_q$ and $h \in \mathbb{G}_2$, output yes if $h = e(P, P)^{abc}$ and no otherwise.

Definition 11 (DBDH Assumption)**.** *The* DBDH *problem is* $\varepsilon_{\mathsf{DBDH}}$*-hard in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if, for all probabilistic polynomial-time algorithms* $\mathcal{A}$*, we have*

$$\begin{aligned}&\mathsf{AdvDBDH}_{(\mathbb{G}_1,\mathbb{G}_2),\mathcal{A}}(k) \doteq \\ &|Pr[\mathcal{A}(P, aP, bP, cP, h) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q, h = e(P, P)^{abc}] - \\ &Pr[\mathcal{A}(P, aP, bP, cP, h) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q, h \overset{R}{\leftarrow} \mathbb{G}_2]| < \varepsilon_{\mathsf{DBDH}}\end{aligned}$$

where the probability is over the random selection of P *from* $\mathbb{G}_1$*, of* a, b, c *from* $\mathbb{Z}_q$*, and over* $\mathcal{A}$*'s random coins.*

The decisional bilinear Diffie-Hellman Inversion (DBDHI) problem in $(\mathbb{G}_1, \mathbb{G}_2)$:

Given $(P, aP, b^2P, \ldots, b^\ell P, h)$ for random $P \in \mathbb{G}_1$, $a, b \in \mathbb{Z}_q$ and $h \in \mathbb{G}_2$, output yes if $h = e(P, P)^{a/b}$ and no otherwise.

Definition 12 (DBDHI Assumption)**.** *The* DBDHI *problem is* $\varepsilon_{\mathsf{DBDHI}}$*-hard in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if, for all probabilistic polynomial-time algorithms* $\mathcal{A}$*, we have*

$$\begin{aligned}&\mathsf{AdvDBDHI}_{(\mathbb{G}_1,\mathbb{G}_2),\mathcal{A}}(k) \doteq \\ &|Pr[\mathcal{A}(P, aP, b^2P, \ldots, b^\ell P, h) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b \overset{R}{\leftarrow} \mathbb{Z}_q, h = e(P, P)^{a/b}] - \\ &Pr[\mathcal{A}(P, aP, b^2P, \ldots, b^\ell P, h) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b \overset{R}{\leftarrow} \mathbb{Z}_q, h \overset{R}{\leftarrow} \mathbb{G}_2]| < \varepsilon_{\mathsf{DBDHI}}\end{aligned}$$

where the probability is over the random selection of P from $\mathbb{G}_1$, of a, b from $\mathbb{Z}_q$, and over $\mathcal{A}$'s random coins.

The modified decisional bilinear Diffie-Hellman (DBDH^1-M) **problem in $\mathbb{G}_1$:**

Given(P, aP, bP, U) for some $a, b \in \mathbb{Z}_q$ and $U \in \mathbb{G}_1$, output yes if $U = ab^2P$ and no otherwise.

Definition 13 (DBDH^1-M Assumption). *The* DBDH^1-M *problem is $\varepsilon_{\mathsf{DBDH}^1\text{-}\mathsf{M}}$-hard in $\mathbb{G}_1$ if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, we have*

$$\begin{aligned}&\mathsf{AdvDBDH}^1\text{-}\mathsf{M}_{\mathbb{G}_1,\mathcal{A}}(k) \doteq \\ &|Pr[\mathcal{A}(P, aP, bP, U) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b \overset{R}{\leftarrow} \mathbb{Z}_q, U = ab^2P]- \\ &\quad Pr[\mathcal{A}(P, aP, bP, U) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b \overset{R}{\leftarrow} \mathbb{Z}_q, U \overset{R}{\leftarrow} \mathbb{G}_1]| < \varepsilon_{\mathsf{DBDH}^1\text{-}\mathsf{M}}\end{aligned}$$

where the probability is over the random selection of P from $\mathbb{G}_1$, of a, b from $\mathbb{Z}_q$, and over $\mathcal{A}$'s random coins.

The extended decisional bilinear Diffie-Hellman (DBDH^2-E) **problem in $\mathbb{G}_2$:**

Given $(P, aP, bP, cP, ab^2P, h)$ for some $a, b, c \in \mathbb{Z}_q$ and $h \in \mathbb{G}_2$, output yes if $h = e(P, P)^{cb^2}$ and no otherwise.

Definition 14 (DBDH^2-E Assumption). *The* DBDH^2-E *problem is $\varepsilon_{\mathsf{DBDH}^2\text{-}\mathsf{E}}$-hard in $\mathbb{G}_2$ if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, we have*

$$\begin{aligned}&\mathsf{AdvDBDH}^2\text{-}\mathsf{E}_{\mathbb{G}_1,\mathcal{A}}(k) \doteq \\ &|Pr[\mathcal{A}(P, aP, bP, cP, ab^2P, h) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q, h = e(P, P)^{ab^2}]- \\ &\quad Pr[\mathcal{A}(P, aP, bP, cP, ab^2P, h) = yes \mid P \overset{R}{\leftarrow} \mathbb{G}_1, a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q, h \overset{R}{\leftarrow} \mathbb{G}_2]| < \varepsilon_{\mathsf{DBDH}^2\text{-}\mathsf{E}}\end{aligned}$$

where the probability is over the random selection of P from $\mathbb{G}_1$, of a, b, c from $\mathbb{Z}_q$, and over $\mathcal{A}$'s random coins.

2.4 Identity-Based Cryptography

2.4.1 Identity-Based Encryption (IBE)

An Identity-Based Encryption scheme [74, 18, 19, 30, 81] is an encryption system in which the identity of each user acts as his/her public key. The corresponding secret key, rather than being directly generated by the individual users, is computed by a trusted central authority (Root), who is also responsible for initially setting up appropriate system-wide parameters.

Initially suggested as a means to alleviate the dependence of public-key encryption systems from the availability of a Public-Key Infrastructure, Identity-Based Encryption schemes have in fact proved to be a powerful primitive whose applications go beyond their original motivation. In particular, they constitutes an important tool for the development of the results presented in Chapter 4 and Chapter 5.

Definition 15 (Identity-Based Encryption Scheme).
An IBE scheme $\mathcal{E}_{\mathsf{IBE}}$ is a four-tuple of probabilistic polynomial-time algorithms (Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$), where:

- Setup is a probabilistic algorithm used by the Root to initialize the parameters of the scheme. Setup takes as input a security parameter 1^λ and returns the global public key $\mathsf{params}_{\mathsf{IBE}}$ and the master secret key $\mathsf{master}_{\mathsf{IBE}}$. Root publishes $\mathsf{params}_{\mathsf{IBE}}$ and keeps $\mathsf{master}_{\mathsf{IBE}}$ secret.

- $\mathsf{Extract}$ takes as input the system parameters $\mathsf{params}_{\mathsf{IBE}}$, the master secret key $\mathsf{master}_{\mathsf{IBE}}$, the identity ID_i of a new user, and returns the corresponding secret key SK_i.

- $\mathsf{Encrypt}$ takes as input the system parameters $\mathsf{params}_{\mathsf{IBE}}$, the recipient's identity ID_i and a message m, and returns a ciphertext C.

- $\mathsf{Decrypt}$ takes as input $\mathsf{params}_{\mathsf{IBE}}$, an identity ID_i, the corresponding secret key SK_i

and a ciphertext C. $\mathsf{Decrypt}$ outputs the hidden message m or a special rejection symbol $\perp$.

An IBE scheme $\mathcal{E}_{\mathsf{IBE}}$ should satisfy the following correctness constraint: for any pair ($\mathsf{params}_{\mathsf{IBE}}$, $\mathsf{master}_{\mathsf{IBE}}$) output by $\mathsf{IBE.Setup}(1^\lambda)$, any secret key SK_i properly generated for identifier ID_i, and any message m:

$$m = \mathsf{IBE.Decrypt}(\mathsf{params}_{\mathsf{IBE}}, \mathrm{ID}_i, \mathsf{SK}_i, \mathsf{IBE.Encrypt}(\mathsf{params}_{\mathsf{IBE}}, \mathrm{ID}_i, m)).$$

An IBE scheme $\mathcal{E}_{\mathsf{IBE}}$ is secure against chosen-ciphertext attack if no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:
Setup: The challenger runs algorithm $\mathsf{IBE.Setup}(1^\lambda)$; it then gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{IBE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{IBE}}$ secret to itself.
Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. Corrupt(ID_i): the challenger runs algorithm $\mathsf{IBE.Extract}(\mathrm{ID}_i)$ to generate the private key SK_i corresponding to user ID_i, and sends SK_i to $\mathcal{A}$.

2. Decrypt(ID_i, C_j): the challenger runs algorithm $\mathsf{IBE.Extract}(\mathrm{ID}_i)$ to recover the private key SK_i corresponding to user ID_i. It then runs $\mathsf{IBE.Decrypt}$($\mathsf{params}_{\mathsf{IBE}}$, ID_i, SK_i, C_j) and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs an identity ID^* and two equal length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue the query Corrupt(ID^*). The challenger picks a random bit $b \in \{0,1\}$, sets $C^* \doteq \mathsf{IBE.Encrypt}(\mathsf{params}_{\mathsf{IBE}}, \mathrm{ID}^*, m_b)$, and sends C^* as a challenge to $\mathcal{A}$.
Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is one of the following:

1. Corrupt(ID_i): the challenger first checks that $\mathrm{ID}_i \neq \mathrm{ID}^*$ and if so, it responds as in **Phase 1**.

2. $\textsc{Decrypt}(\mathrm{ID}_i, C_j)$: the challenger first checks that $C_j \neq C^*$ and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0, 1\}$ and wins the game if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as an IBE.IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}_{\mathsf{IBE}}$ as:

$$\mathsf{Adv}_{\mathsf{IBE},\mathcal{A}} = \left|\Pr[b = b^*] - \frac{1}{2}\right| .$$

The probability is over the random bits of the challenger and of the adversary.

A weaker notion of security, initially introduced by Canatti, Halevi and Katz [26], requires the adversary to commit ahead of time to the identity it will attack. We refer to this notion as selective-identity chosen-ciphertext security (IBE.IND-sID-CCA). The game is exactly as for the IBE.IND-ID-CCA case, except that the adversary $\mathcal{A}$ discloses to the challenger the target identity ID^* *before* the **Setup** phase. Thus, the restriction on $\textsc{Corrupt}$ queries from **Phase 2** also holds in **Phase 1**.

Definition 16 ($\textsc{Chosen-Ciphertext Security for}$ IBE)**.**
An IBE scheme $\mathcal{E}_{\mathsf{IBE}}$ is $(\tau, q_{\mathrm{ID}}, q_C, \varepsilon_{\mathsf{IBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure if, for any τ-time IBE.IND-ID-CCA (resp. IBE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen $\textsc{Corrupt}$ queries and at most q_C chosen $\textsc{Decrypt}$ queries, it holds that $\mathsf{Adv}_{\mathsf{IBE},\mathcal{A}} < \varepsilon_{\mathsf{IBE}}$.

We define chosen-plaintext security for an IBE scheme according to an attack game that is defined exactly as described above, except that the adversary is not allowed to issue $\textsc{Decrypt}$ queries. Notice, however, that the adversary may still issue adaptive chosen $\textsc{Corrupt}$ queries. This security notion is termed IBE.IND-ID-CPA (or IBE.IND-sID-CPA in the case of selective identity adversary).

Definition 17 ($\textsc{Chosen-Plaintext Security for}$ IBE)**.**
An IBE scheme $\mathcal{E}_{\mathsf{IBE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{IBE}})$-IND-ID-CPA (resp. IND-sID-CPA)-secure if $\mathcal{E}_{\mathsf{IBE}}$ is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{IBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure.

2.4.2 Hierarchical Identity-Based Encryption (HIBE)

A Hierarchical Identity-Based Encryption scheme [52, 48, 16] is a natural extension of the notion of Identity-Based Encryption scheme. Intuitively, HIBE allows to organize the users into a tree hierarchy. Each user gets the secret key from its parent in the hierarchy (and all the users share a few global parameters, initially set up by the Root). Then, anybody can encrypt messages to any given user knowing just *its position in the hierarchy*, specified as an ID-tuple (or *hierarchical* identifier) $\mathrm{HID} \equiv (\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$. This means that if we consider a user located at level i, all its ancestors, starting from its parent up to the Root, have hierarchical identifiers $(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-1})$, $(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-2})$, …, (ID_1), Root.

Definition 18 (Hierarchical Identity-Based Encryption Scheme)**.**
An HIBE scheme $\mathcal{E}_{\mathsf{HIBE}}$ is a four-tuple of probabilistic polynomial-time algorithms (Setup, Extract, Encrypt, Decrypt), where:

- Setup takes as input a security parameter 1^λ and returns the global public key $\mathsf{params}_{\mathsf{HIBE}}$ and the master secret key $\mathsf{master}_{\mathsf{HIBE}}$. Root publishes $\mathsf{params}_{\mathsf{HIBE}}$ and keeps $\mathsf{master}_{\mathsf{HIBE}}$ secret.

- Extract takes as input the system parameters $\mathsf{params}_{\mathsf{HIBE}}$, an identity ID-tuple $\mathrm{HID}_i \equiv (\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ (with the convention that $\mathrm{HID}_0 \equiv$ Root) and the corresponding secret key SK_i (with the convention that $\mathsf{SK}_0 \equiv \mathsf{master}_{\mathsf{HIBE}}$). It returns the secret key SK_{i+1} ID-tuple $\mathrm{HID}_{i+1} \equiv (\mathrm{ID}_1, \ldots, \mathrm{ID}_i, \mathrm{ID}_{i+1})$.

- Encrypt takes as input the system parameters $\mathsf{params}_{\mathsf{HIBE}}$, the recipient's ID-tuple HID_i and a message m, and returns a ciphertext C.

- Decrypt takes as input $\mathsf{params}_{\mathsf{HIBE}}$, an ID-tuple HID_i, the corresponding secret key SK_i and a ciphertext C. Decrypt outputs the hidden message m or a special rejection symbol $\perp$.

An HIBE scheme $\mathcal{E}_{\mathsf{HIBE}}$ should satisfy the following correctness constraint: for any pair ($\mathsf{params}_{\mathsf{HIBE}}$, $\mathsf{master}_{\mathsf{HIBE}}$) output by $\mathsf{HIBE.Setup}(1^\lambda)$, any secret key SK_i properly generated for identifier HID_i, and any message m:

$$m = \mathsf{HIBE.Decrypt}(\mathsf{params}_{\mathsf{HIBE}}, \mathrm{HID}_i, \mathsf{SK}_i, \mathsf{HIBE.Encrypt}(\mathsf{params}_{\mathsf{HIBE}}, \mathrm{HID}_i, m)).$$

We notice that in the case of HIBE, all the ancestors of a given user can recover the messages encrypted for this user, *e.g.* by first deriving the secret key for the descendant with a sequence of $\mathsf{Extract}$ operations, and then decrypting the ciphertext. For specific schemes, however, there might be more efficient/direct ways to perform such decryption. For example, the HIBE of [48] enjoys a more efficient decryption by any ancestor of the given node than by the node itself.

An HIBE scheme $\mathcal{E}_{\mathsf{HIBE}}$ is secure against chosen-ciphertext attack if no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

Setup: The challenger runs algorithm $\mathsf{HIBE.Setup}(1^\lambda)$; it gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{HIBE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{HIBE}}$ secret to itself.

Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. $\textsc{Corrupt}(\mathrm{HID}_i)$: starting from $\mathsf{master}_{\mathsf{HIBE}}$, the challenger uses the $\mathsf{HIBE.Extract}$ algorithm to derive the secret key corresponding to all the ancestors of HID_i, eventually obtaining the secret key SK_i, which is then sent to $\mathcal{A}$.

2. $\textsc{Decrypt}(\mathrm{HID}_i, C_j)$: the challenger runs algorithm $\mathsf{HIBE.Extract}(\mathrm{HID}_i)$ to recover the private key SK_i corresponding to user HID_i. It then runs algorithm $\mathsf{HIBE.Decrypt}(\mathsf{params}_{\mathsf{HIBE}}$, HID_i, SK_i, $C_j)$ and sends the result to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs an identity HID^* and two equal length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue $\textsc{Corrupt}(\cdot)$ queries on HID^*

or a prefix of HID^*. The challenger picks a random bit $b \in \{0,1\}$, sets $C^* \doteq \mathsf{HIBE.Encrypt}(\mathsf{params}_{\mathsf{HIBE}}, \mathrm{HID}^*, m_b)$, and sends C^* as a challenge to $\mathcal{A}$.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is one of the following:

1. Corrupt(HID_i): the challenger first checks that $\mathrm{HID}_i \neq \mathrm{HID}^*$ and that HID_i is not a prefix of HID^*, and if both conditions are satisfied it responds as in **Phase 1**.

2. Decrypt(HID_i, C_j): the challenger first checks that either $C_j \neq C^*$ or HID_i is not HID^* nor a prefix of HID^*, and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0,1\}$ and wins the game if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as an HIBE.IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}_{\mathsf{HIBE}}$ as:

$$\mathsf{Adv}_{\mathsf{HIBE},\mathcal{A}} = \left| \Pr[b = b^*] - \frac{1}{2} \right| .$$

The probability is over the random bits of the challenger and of the adversary.

Similarly to the case of a regular IBE scheme, we can also define the notion of selective-identity chosen-ciphertext security for a HIBE (HIBE.IND-sID-CCA). The game is exactly as for the HIBE.IND-ID-CCA case, except that the adversary $\mathcal{A}$ discloses to the challenger the target identity HID^* *before* the **Setup** phase. Thus, the restriction on Corrupt queries from **Phase 2** also holds in **Phase 1**.

Definition 19 (Chosen-Ciphertext Security for HIBE)**.**
An HIBE scheme $\mathcal{E}_{\mathsf{HIBE}}$ is $(\tau, q_{\mathrm{ID}}, q_C, \varepsilon_{\mathsf{HIBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure if for any τ-time HIBE.IND-ID-CCA (resp. HIBE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen Corrupt queries and at most q_C chosen Decrypt queries, it holds that $\mathsf{Adv}_{\mathsf{HIBE},\mathcal{A}} < \varepsilon_{\mathsf{HIBE}}$.

We define chosen-plaintext security for a HIBE scheme according to an attack game that is exactly as in the preceding game, except that the adversary is not allowed to issue Decrypt queries. The adversary may still issue adaptive chosen Corrupt queries.

Definition 20 (Chosen-Plaintext Security for HIBE).
An HIBE scheme $\mathcal{E}_{\mathsf{HIBE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{HIBE}})$-IND-ID-CPA (resp. IND-sID-CPA)-secure if $\mathcal{E}_{\mathsf{HIBE}}$ is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{HIBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure.

2.4.3 Paired Hierarchical Identity-Based Encryption (PHIBE)

Paired Hierarchical Identity-Based Encryption [82] is a generalization of Hierarchical Identity-Based Encryption to the case where each entity occupies a place in *two* independent hierarchies. In a standard HIBE, an entity is specified by a hierarchical identifier HID, *i.e.* a sequence of identifiers; in a PHIBE, an entity is specified by a *pair* of hierarchical identifiers $\mathrm{PID} \doteq \langle \mathrm{HID}^{\ell}, \mathrm{HID}^{r} \rangle$, one for the *left* hierarchy and the other for the *right* hierarchy. As shown in [82], it is possible to extend this idea to any (constant) number of independent hierarchies.

In the context of PHIBE, an entity $\mathrm{PID}_1 = \langle \mathrm{HID}_1^{\ell}, \mathrm{HID}_1^{r} \rangle$ is considered an ancestor of $\mathrm{PID}_2 = \langle \mathrm{HID}_2^{\ell}, \mathrm{HID}_2^{r} \rangle$ if HID_1^{ℓ} is a prefix of HID_2^{ℓ} *and* HID_1^{r} is a prefix of HID_2^{r} (*i.e.* if PID_1 is an ancestor of PID_2 in *both* hierarchies.)

Definition 21 (Paired Hierarchical Identity Based Encryption Scheme).
A PHIBE scheme $\mathcal{E}_{\mathsf{PHIBE}}$ is a four-tuple of probabilistic polynomial-time algorithms (Setup, Extract, Encrypt, Decrypt), where:

- Setup takes as input 1^{λ} and $1^{m^{\ell}}, 1^{m^{r}}$ (where λ is the security parameter and m^{ℓ}, m^{r} are the maximum depth of the left and right hierarchies, respectively), and generates some global parameters $\mathsf{params}_{\mathsf{PHIBE}}$ and the master secret key $\mathsf{master}_{\mathsf{PHIBE}}$. Root publishes $\mathsf{params}_{\mathsf{PHIBE}}$ and keeps $\mathsf{master}_{\mathsf{PHIBE}}$ secret.

- Extract is a probabilistic algorithm run by an entity PID_1 to generate the secret key for a descendant PID_2. Extract takes as input $\mathsf{params}_{\mathsf{PHIBE}}$, the paired hierarchical identifier PID_2, the secret key $\mathsf{SK}_{\mathrm{PID}_1}$ and outputs $\mathsf{SK}_{\mathrm{PID}_2}$.

- Encrypt takes as input the system parameters $\mathsf{params}_{\mathsf{PHIBE}}$, the recipient's paired hierarchical identifier PID_i and a message m, and returns a ciphertext C.

- $\mathsf{Decrypt}$ takes as input $\mathsf{params}_{\mathsf{PHIBE}}$, a paired hierarchical identifier PID_i, the corresponding secret key SK_i and a ciphertext C. $\mathsf{Decrypt}$ outputs the hidden message m or a special rejection symbol $\perp$.

A PHIBE scheme $\mathcal{E}_{\mathsf{PHIBE}}$ should satisfy the following correctness constraint: for any pair $(\mathsf{params}_{\mathsf{PHIBE}}, \mathsf{master}_{\mathsf{PHIBE}})$ output by $\mathsf{PHIBE.Setup}(1^\lambda)$, any secret key SK_i properly generated for identifier PID_i, and any message m:

$$m = \mathsf{PHIBE.Decrypt}(\mathsf{params}_{\mathsf{PHIBE}}, \mathrm{PID}_i, \mathsf{SK}_i, \mathsf{PHIBE.Encrypt}(\mathsf{params}_{\mathsf{PHIBE}}, \mathrm{PID}_i, m)).$$

A PHIBE scheme $\mathcal{E}_{\mathsf{PHIBE}}$ is secure against chosen-ciphertext attack if no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

Setup: The challenger runs algorithm $\mathsf{PHIBE.Setup}(1^\lambda, 1^{m^\ell}, 1^{m^r})$; it gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{PHIBE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{PHIBE}}$ secret.

Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. $\textsc{Corrupt}(\mathrm{PID}_i)$: starting from $\mathsf{master}_{\mathsf{PHIBE}}$, the challenger uses $\mathsf{PHIBE.Extract}$ to derive the secret key corresponding to all the ancestors of PID_i, eventually obtaining the secret key SK_i, which is then sent to $\mathcal{A}$.

2. $\textsc{Decrypt}(\mathrm{HID}_i, C_j)$: the challenger runs algorithm $\mathsf{PHIBE.Extract}(\mathrm{PID}_i)$ to recover the private key SK_i corresponding to user PID_i. It then runs algorithm $\mathsf{PHIBE.Decrypt}(\mathsf{params}_{\mathsf{PHIBE}}, \mathrm{PID}_i, \mathsf{SK}_i, C_j)$ and sends the result to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs an identity PID^* and two equal length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue the $\textsc{Corrupt}(\cdot)$ queries on PID^* or a prefix of PID^*. The challenger picks a random bit $b \in \{0,1\}$, sets $C^* \doteq \mathsf{PHIBE.Encrypt}(\mathsf{params}_{\mathsf{PHIBE}}, \mathrm{PID}^*, m_b)$, and sends C^* as a challenge to $\mathcal{A}$.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is one of the following:

1. Corrupt(PID_i): the challenger first checks that $\mathrm{PID}_i \neq \mathrm{PID}^*$ and that PID_i is not a prefix of PID^*, and if both conditions are satisfied, it responds as in **Phase 1**.

2. Decrypt(PID_i, C_j): the challenger first checks that either $C_j \neq C^*$ or PID_i is not PID^* nor a prefix of PID^*, and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0,1\}$ and wins the game if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as a PHIBE.IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}_{\mathsf{PHIBE}}$ as:

$$\mathsf{Adv}_{\mathsf{PHIBE},\mathcal{A}} = \left|\Pr[b = b^*] - \frac{1}{2}\right|.$$

The probability is over the random bits of the challenger and of the adversary.

Similarly to the case of a regular HIBE scheme, we can also define the notion of selective-identity chosen-ciphertext security for PHIBE (PHIBE.IND-sID-CCA). The game is exactly as for the PHIBE.IND-ID-CCA case, except that the adversary $\mathcal{A}$ discloses to the challenger the target identity PID^* *before* the **Setup** phase. Thus, the restriction on Corrupt queries from **Phase 2** also holds in **Phase 1**.

Definition 22 (Chosen-Ciphertext Security for PHIBE).
A PHIBE scheme $\mathcal{E}_{\mathsf{PHIBE}}$ is $(\tau, q_{\mathrm{ID}}, q_C, \varepsilon_{\mathsf{PHIBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure if for any τ-time PHIBE.IND-ID-CCA (resp. PHIBE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen Corrupt queries and at most q_C chosen Decrypt queries, it holds that $\mathsf{Adv}_{\mathsf{PHIBE},\mathcal{A}} < \varepsilon_{\mathsf{PHIBE}}$.

We define chosen-plaintext security for an PHIBE scheme according to an attack game that is exactly as the game described above, except that the adversary is not allowed to issue Decrypt queries. The adversary may still issue adaptive chosen Corrupt queries.

Definition 23 (Chosen-Plaintext Security for PHIBE).
An PHIBE scheme $\mathcal{E}_{\mathsf{PHIBE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{PHIBE}})$-IND-ID-CPA (resp. IND-sID-CPA)-secure if $\mathcal{E}_{\mathsf{PHIBE}}$ is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{PHIBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure.

2.5 Forward-Secure Encryption

The central idea of forward secrecy is that the compromise of long-term keys does not compromise past session keys and therefore past communications. This notion was first proposed by Günther [47] and later by Diffie *et al.* [33] in key exchange protocols.

The notion of non-interactive forward security was proposed by Anderson [3] in 1997 and later formalized by Bellare and Miner [6], who also gave a forward-secure signature scheme followed by a line of improvement [1, 64]. In this model, secret keys are updated at regular intervals throughout the lifetime of the system; furthermore, exposure of a secret key corresponding to a given interval does not enable an adversary to break the system (in the appropriate sense) for any prior time period. The model inherently cannot prevent the adversary from breaking the security of the system for any subsequent time period. Bellare and Yee [7] provided a comprehensive treatment of forward security in the context of private key based cryptographic primitives.

The first forward-secure public-key encryption scheme was given by Canetti, Halevi, and Katz [26] based on the Gentry-Silverberg HIBE scheme [48]. The scheme constructs a binary tree, in which a tree node corresponds to a time period and has a secret key. Children of a node w are labeled $w0$ and $w1$, respectively. Given the secrets corresponding to a prefix of a node representing time t, one can compute the secrets of time t. In order to make future keys computable from the current key, the secrets associated with a prefix of a future time are stored in the current key. After the key for the next time period is generated, the current decryption key is erased.

Definition 24 (Forward-Secure Encryption Scheme)**.**

A FSE scheme $\mathcal{E}_{\mathsf{FSE}}$ is a four-tuple of probabilistic polynomial-time algorithms (Setup, Update, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$), where:

- Setup is used by the Root to initialize the parameters of the scheme. Setup takes as input a security parameter 1^λ and the total number of time periods T. It returns the global public key $\mathsf{params}_{\mathsf{FSE}}$ and the master secret key $\mathsf{master}_{\mathsf{FSE}}$. Root publishes $\mathsf{params}_{\mathsf{FSE}}$ and keeps $\mathsf{master}_{\mathsf{FSE}}$ secret.

- Update takes as input the system parameters $\mathsf{params}_{\mathsf{FSE}}$, the current time period $t < T$ and the user's secret key SK_t. It outputs the new secret key SK_{t+1} valid for the following time period $t + 1$.

- $\mathsf{Encrypt}$ takes as input the system parameters $\mathsf{params}_{\mathsf{FSE}}$, the current time period t and a message m and returns the ciphertext C.

- $\mathsf{Decrypt}$ takes as input a time period $t < T$, the corresponding secret key SK_t and a ciphertext C. $\mathsf{Decrypt}$ outputs the hidden message m or a special rejection symbol $\perp$.

A FSE scheme $\mathcal{E}_{\mathsf{FSE}}$ should satisfy the following correctness constraint: for any pair ($\mathsf{params}_{\mathsf{FSE}}$, $\mathsf{master}_{\mathsf{FSE}}$) output by $\mathsf{FSE.Setup}(1^\lambda, T)$, any time instant $t < T$ and secret key SK_t properly generated for t, and any message m:

$$m = \mathsf{FSE.Decrypt}(\mathsf{params}_{\mathsf{FSE}}, t, \mathsf{SK}_t, \mathsf{FSE.Encrypt}(\mathsf{params}_{\mathsf{FSE}}, t, m)).$$

A FSE scheme $\mathcal{E}_{\mathsf{FSE}}$ is secure against chosen-ciphertext attack if no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:
Setup: The challenger runs algorithm $\mathsf{FSE.Setup}(1^\lambda, T)$; it then gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{FSE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{FSE}}$ secret to itself.
Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. CORRUPT(t): the challenger repeatedly runs algorithm $\mathsf{FSE.Update}(\mathsf{params}_{\mathsf{FSE}}, t, \cdot)$ to generate the private key SK_t corresponding to time instant t, and sends SK_t to $\mathcal{A}$.

2. DECRYPT(t, C_j): the challenger first computes the private key SK_t corresponding to time instant t. It then runs algorithm $\mathsf{FSE.Decrypt}$($\mathsf{params}_{\mathsf{FSE}}$, t, SK_t, C_j) and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs a time instant t^* and two equal length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue the query $\textsc{Corrupt}(t)$, for $0 \leq t < t^* \leq T$. The challenger picks a random bit $b \in \{0, 1\}$, sets $C^* \doteq \mathsf{FSE.Encrypt}(\mathsf{params}_{\mathsf{FSE}}, t^*, m_b)$, and sends C^* as a challenge to $\mathcal{A}$.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is one of the following:

1. $\textsc{Corrupt}(t)$: the challenger first checks that $0 \leq t^* < t \leq T$ and if so, it responds as in **Phase 1**.

2. $\textsc{Decrypt}(t, C_j)$: the challenger first checks that $C_j \neq C^*$ and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0, 1\}$ and wins the game if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as an FSE.IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}_{\mathsf{FSE}}$ as:

$$\mathsf{Adv}_{\mathsf{FSE},\mathcal{A}} = \left| \Pr[b = b^*] - \frac{1}{2} \right| .$$

The probability is over the random bits of the challenger and of the adversary.

In [26], Canetti *et al.* introduced a weaker notion of security, which requires the adversary to commit ahead of time to the time instant it will attack. We refer to this notion as selective-identity chosen-ciphertext security (FSE.IND-sID-CCA). The game is exactly as for the FSE.IND-ID-CCA case, except that the adversary $\mathcal{A}$ discloses to the challenger the target instant t^* *before* the **Setup** phase. Thus, the restriction on $\textsc{Corrupt}$ queries from **Phase 2** also holds in **Phase 1**.

Definition 25 ($\textsc{Chosen-Ciphertext Security for}$ FSE)**.**
A FSE scheme $\mathcal{E}_{\mathsf{FSE}}$ is $(\tau, q_{\mathrm{ID}}, q_C, \varepsilon_{\mathsf{FSE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure if, for any τ-time FSE.IND-ID-CCA (resp. FSE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen $\textsc{Corrupt}$ queries and at most q_C chosen $\textsc{Decrypt}$ queries, it holds that $\mathsf{Adv}_{\mathsf{FSE},\mathcal{A}} < \varepsilon_{\mathsf{FSE}}$.

We define chosen-plaintext security for a FSE scheme according to an attack game that is defined exactly as described above, except that the adversary is not allowed to issue DECRYPT queries. Notice, however, that the adversary may still issue adaptive chosen CORRUPT queries. This security notion is termed FSE.IND-ID-CPA (or FSE.IND-sID-CPA in the case of selective identity adversary).

Definition 26 (CHOSEN-PLAINTEXT SECURITY FOR FSE)**.**
A FSE scheme $\mathcal{E}_{\mathsf{FSE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{FSE}})$-IND-ID-CPA (resp. IND-sID-CPA)-secure if $\mathcal{E}_{\mathsf{FSE}}$ is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{FSE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure.

Chapter 3

DRM-Enabling Crypto Primitives and Previous Constructions

3.1 Broadcast Encryption

A *broadcast encryption* scheme allows the sender to securely distribute data to a dynamically changing set of users over an insecure channel. Namely, it should be possible to selectively *revoke* (*i.e.* exclude) a certain subset of users from receiving the data. For that reason, broadcast encryption schemes are often also referred to as *revocation schemes*, since the revocation ability is what makes the task of broadcast encryption non-trivial. In particular, each user should receive an individualized decoder (*i.e.* a decryption device with a unique secret key) which decrypts only the ciphertexts intended for the given user. The relevance of broadcast encryption for DRM technologies stems from its applicability to access control, which enables applications such as pay-TV systems, distribution of copyrighted material, streaming audio/video and many others.

A broadcast encryption scheme consists of the following three algorithms:

- **key assignment algorithm**: every new user joining the system receives a personalized secret information, needed to recover the content from the broadcast;

- **broadcast algorithm**: given in input the set of privileged users and the message

$\mathcal{M}$, outputs the encryption of $\mathcal{M}$ to be sent in broadcast;

- **decryption algorithm**: given in input the encrypted content and the secret information of a user, output the corresponding message $\mathcal{M}$ if the user is authorized to receive the message and nonsense otherwise.

The above description of a broadcast encryption scheme is intentionally vague, and can be refined into many different variants, depending on the specific scenario under consideration. To name just a few, the scheme can be secret-key or public-key based; it can support a single, bounded or unbounded number of broadcasts; it might support a bounded or unbounded number of revocations per broadcast; the set of receivers can be fixed, slowly changing or rapidly changing; it might or might not be possible to periodically refresh users' secret keys (*stateful vs. stateless* receivers setting).

For ease of comparison, before starting analyzing existing solutions, we fix some notation. Let $\mathcal{N}$ be the set of all users, with $|\mathcal{N}| = N$. Let $\mathcal{R}$ be the set of revoked users, with $|\mathcal{R}| = r$ and denote with c the upper bound on the maximum coalition size. In some cases, the efficiency of the scheme will depend just on c, and in others just on r (in which case, we may think $c = r$).

The efficiency of a broadcast encryption scheme is evaluated considering the following parameters:

- **Storage complexity** (κ_s): denotes the amount of personalized information each user needs to store to recover the content from the broadcast;

- **Communication complexity** (κ_c): denotes the length of the broadcast;

- **Encryption/Decryption complexity** (κ_e/κ_d): measure the running time to create the ciphertext corresponding to the message, and to recover the content from the broadcast, respectively.

3.1.1 Early Approaches

A naïve solution to the problem of broadcast encryption consists of assigning a personalized secret key to each user and then having the transmitter sending individualized messages to each recipient. This solution is clearly unsatisfactory, as it entails communication and encryption complexity linear in $N - r$.

Another trivial approach is to assign a secret key to each possible subset of $\mathcal{N}$ and then give each user the secret keys corresponding to all subsets to which he belongs. After such initialization phase, the transmitter can broadcast to any subset of users, with just a single ciphertext, but the storage complexity at each recipient is exponential in N.

In [10], Berkovits proposed a broadcast scheme that, although requiring communication, encryption and decryption complexity linear in the number of recipients $N - r$, introduces an approach largely employed (in a revised form) in subsequent work.

The scheme of [10] is based on Shamir's "k out of m" secret sharing scheme [73]: a secret s from a finite field $\mathbb{F}_p$ is encoded as the constant term of some polynomial P over $\mathbb{F}_p$ of degree k, and each participant gets a "share" consisting of the value of P at some point x_i (where the points assigned to all the users are assumed to be pairwise distinct). Any $k + 1$ participants can then combine their shares to recover s via interpolation.

The idea of the scheme of [10] is to assign a pair $(x_i, y_i) \in \mathbb{F}_p^2$ as secret key for user $i = 1, \ldots, N$. To transmit a message s, the sender selects a random polynomial P of degree k (in our notation, $k \doteq N - r$), having s as constant coefficient and passing through the points corresponding to the authorized users but not through the points of the revoked users. Then, the sender broadcasts k shares of P corresponding to points not assigned to the participants. Each authorized subscriber adds its key y_i to the received shares and, using Shamir's secret sharing, recovers the secret s, whereas the secret remains still random in the view of revoked users.

The merit of the scheme lies in that it shows that it is possible to encode a message so that each and every listener needs to process the entire broadcast to recover the plaintext, and guarantees that each authorized recipient deduces the secret, while all others deduce nonsense.

The formal study of broadcast encryption was initiated by Fiat and Naor [43], who show a scheme with storage complexity $O(c \log c \log N)$ and communication complexity $O(c^2 \log^2 c \log N)$, regardless of the actual number of revoked users r. In other words, the scheme allows to broadcast to any subset of users, guaranteeing that no coalition of at most c users can recover the content. They also present a probabilistic scheme, with storage complexity $O(\log c \log(1/\alpha))$ and communication complexity $O(c \log^2 c \log(1/\alpha))$, in which security against a subset of c revoked users holds with probability $1 - \alpha$.

3.1.2 Information Theoretic Lower Bounds

The works of Blundo and Cresti [14] and of Blundo *et al.* [15], represent early efforts in investigating the limits and trade-offs intrinsic to the problem of broadcast encryption.

Along the same line of research, in [62] Luby and Staddon employ combinatorial techniques to derive a trade-off between the storage and the communication complexities in an *information-theoretic* model, in which the storage is measured in terms of the actual number of secret keys that each user needs to decrypt the broadcast. Such information-theoretic model rules out schemes in which decryption keys are derived (in a computationally-secure way) from the secret personalized information that each user receives when he joins the system (*computational* model). For every given upper bound on the information-theoretic storage complexity, they prove lower bounds on the communication complexity. The authors identify two natural classes of protocols that the sender may use in establishing a session key under which the message to broadcast is encrypted: *OR protocols* and *AND protocols*. In an OR protocol, the broadcast consists of several parts, and each recipients only needs to understand one of such components to recover the session key s. In an AND protocol, instead, each user needs to understand all the parts that make up the broadcast in order to infer s. For both classes of protocols, the authors show that a broadcast encryption scheme with communication complexity κ_c must impose an average storage complexity of $\kappa_s \geq \frac{\binom{N}{r}^{1/\kappa_c}}{8r\kappa_c}$. Additionally, for the OR protocols they show that the bound is essentially tight, by providing a simple construction in which the number of keys stored by each user is $\left(\frac{\binom{N}{r}^{1/\kappa_c}}{c} - 1\right)/r$.

3.1.3 Beyond the Lower Bounds

Subsequent research has focused on eluding such lower bounds by twisting the underlying model. This is the case, for example, for the multicast encryption setting, where participants are required to update their secret information each time the set of privileged users changes (*stateful setting*).

The scheme of Kumar *et al.* [59] also beats the lower bound proved in [62], by restricting the problem of broadcast encryption to the case that a *single* message has to be broadcast (also known as the *blacklisting problem*). In particular, [59] provides two constructions with no computational assumptions, that achieve communication complexity $O(r \log N)$ and storage complexity respectively $O(r \log N)$ and $O(r^2)$.

Another line of development was suggested by Garay *et al.* in [46] where, to cope with the growth of the total number of revoked users during the lifetime of a broadcast encryption scheme and the consequential degradation of the system performances, the authors introduce the notion of *long-lived broadcast encryption.* The idea is to give each participant a "card" containing a set of keys, where the cards of two distinct users are not necessarily disjoint. Whenever a user needs to be removed from the system (either because his contract runs out or because he is discovered as a traitor), all the keys contained in his card get revoked; such keys are also crossed out from those cards (belonging to legal users) that overlap with the card to be removed. As more cards get revoked, it may happen that the fraction of keys crossed out in a legal card approaches a threshold above which the card is no longer functional: such card is called *dead.* Dead cards are not replaced immediately, however; rather, the sender uses unicast to deliver the content to the holders of dead cards, up until the time there are sufficiently many dead cards to justify rekeying, When rekeying is deemed appropriate, a new epoch starts and the sender replaces not only the dead cards, but also those "close to die." Because long-lived broadcast encryption aims at improving performance of the system in the long run, the efficiency of schemes designed in such model is better described by a somehow different parameterization than the one we have been looking at so far. In particular, the authors suggest to consider the trade-off between the number of keys per card (which is analogous to the storage complexity) and the expected number of dead cards that can be tolerated before a specific user needs to be recarded (which in a sense is an extension of the communication complexity). Three concrete schemes are presented in [46]: all of them require $\Omega((r \log N + \log 1/\epsilon) \log 1/\gamma)$ keys per card and can tolerate on average $O(r)$ dead cards before require recarding, where γ is a measure of the overlap that is allowed between users' cards, and ϵ is an error probability governing the security of the construction.

The schemes described so far are based on tools from Combinatorics and symmetric-key Cryptography. The literatures also contains broadcast encryption schemes which employ public-key techniques, but since they also provide additional functionalities, we will treat them separately in Section 3.3. Here we observe that when using asymmetric encryption, the storage requirement on the end users is effectively split into storage of *public* information and storage of *private* information. Most existing solutions strive to minimize both quantities at the same (although the bias is always toward the least private storage when a trade-off is inevitable). The recent work of Boneh *et al.* [20] instead shows that, if one can maintain at each end user a very large public storage (proportional to the size of the entire user population), then it is possible to obtain broadcast encryption schemes with *constant* communication complexity and *constant* private user storage, at the cost of having encryption/decryption complexities proportional to the number of recipients.

An interesting variant of the broadcast encryption problem has recently been proposed by Barth *et al.* in [4]. In line with the growing public awareness of the privacy issues related to electronic communication, the authors of [4] argue that it may be desirable to protect not only the content of the broadcast, but also the *recipients' identities*. The combination of these two properties gives rise to what the authors of [4] term a *private* broadcast encryption scheme. As proof-of-concept, Barth *et al.* also suggests a generic and a number-theoretic construction with communication complexity linear in the number of recipients.

3.2 Traitor Tracing

Traitor tracing schemes constitute a very useful tool against piracy in the context of digital content broadcast. They are multi-recipient encryption schemes that can be employed by content providers that wish to deliver copyrighted material to an exclusive set of users. Each user holds a decryption key that is fingerprinted and bound to his identity. In the case of a leakage, a specialized traitor tracing algorithm is run to uncover its source. Therefore, a traitor tracing scheme deters subscribers of a distribution system from leaking information by the mere fact that the identities of the leaking entities (the *traitors*) can be revealed.

A traitor tracing scheme consists of the following four algorithms:

- **key assignment algorithm**: used in the setup phase by the system manager to assign fingerprinted secret information to each user, that enables the user to decrypt, but at the same time is traceable in case of user misbehavior;
- **encryption algorithm**: run by the data supplier to create a ciphertext from the corresponding message;
- **decryption algorithm**: given in input the encrypted content and the secret key of a user, output a possibly personalized variant of the original message;
- **tracing algorithm**: executed by the system manager in case that a leakage is discovered, to expose the identities of the traitors.

To be more precise in the formalization of a traitor tracing scheme it is necessary to define the key characteristics of the setting under analysis. The most important trait to specify is the kind of piracy strategy that the system is supposed to thwart. Two natural strategies that can be employed by the pirates to enable illegal mass-access to the copyrighted material are: (1) leaking the decryption keys, and (2) leaking the decrypted content. Consequently, traitor tracing proposals can be categorized according to the piracy strategy that they aim to contrast.

3.2.1 Tracing Based on Leaked Decryption Keys

Consider the case of a group of users (the traitors) that subscribe to the system with the goal of extracting the decryption keys from their decoders and combine all their secret-key material to obtain an illegal decryption device (the *pirate decoder*) that is then sold on the black-market. If the encryption scheme used in the system assigns the same key to all the users, then such a strategy would be completely risk-free for the pirates, since their identities are totally unrelated to the decryption key within the illegal box. Even if decryption keys are fingerprinted and bound to user identities, the pirates may still hope be able to break the correspondence between keys and users by combining their keys together to obtain a working decryption key that is different from all their own keys. An appealing feature of this kind of piracy (from the pirates' point of view) is that it only requires an initial effort, after which all the content distributed by the system will be available in the black-market at no extra cost. For these reasons, this piracy strategy is the most likely to arise, and it has been considered extensively in the cryptographic literature. The goal is to make sure that when a pirate decoder is discovered, the authorities can trace the identities of the users that contributed in its construction by employing the traitor tracing algorithm.

The first formal definition of traitor tracing scheme appears in Chor *et al.* [28, 29]. To measure the resiliency of a system to collusion of traitors, the authors introduce the notion of *c-traceability*:[1] intuitively, a scheme is c-traceable if, given access to a pirate decoder built by combining at most c decryption keys, it is possible to trace in polynomial time at least one of the traitors that contributed in its construction. In [28], the authors presents several constructions, all based on 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 best of their constructions can guarantee c-traceability, while requiring storage and decryption complexity $O(c^2 \log^2 c \log(N/c))$ and communication complexity $O(c^3 \log^4 c \log(N/c))$. In fact, the authors also consider other solutions that achieve better parameters by assuming that not only the secret keys assigned to the user, but also some of the design choices are kept secret. Under this assumption, the authors provide a scheme that, for any $0 < \alpha < 1$, provides c-traceability with probability $1 - \alpha$, and requires storage and decryption complexity $O(\log(N/\alpha)\log(1/\alpha))$ and communication complexity $O(c \log(N/\alpha)\log(1/\alpha))$.

Explicit combinatorial constructions of traitor tracing schemes were later implemented by Stinson and Wei in [77]. The efficiency of the resulting schemes is asymptotically worse than those in [28], but for practical, small values of c and N, they may actually be better. The authors additionally exploit the connection between traitor tracing schemes and other combinatorial objects to translate known lower bounds from combinatorics to the traitor tracing setting.

Typical applications of traitor tracing are in the entertainment industry, where piracy is only a concern in the case the illegal device is capable of correctly decrypting the entire scrambled content. Indeed, pirates would arguably have few customers in the black-market if the stream recovered by illegal boxes had 10% of each frame blanked out. Moving from this observation, in [66, 29] the authors consider the notion of *threshold* traitor tracing scheme, where the tracing algorithm is only required to guarantee exposure of the traitors' identities for pirate decoders whose decryption probability is better

[1]In [28], the authors actually use the term *resiliency*, rather than traceability. Subsequent work has adopted the latter terminology and we also prefer it here to avoid confusion with resiliency in the context of broadcast encryption.

than a given threshold β. (In the model of [28], successful tracing is required for any illegal box that decrypts with non-negligible probability.) Thanks to such relaxation, the schemes presented in [66] are more efficient than the fully traceable counterparts in [28]; in particular, one construction achieves storage complexity $O(c/\beta \log(N/\alpha))$, communication complexity linear in the maximum size c of coalition and constant decryption complexity. They also show how to improve the storage per user by a factor of roughly $k/\log k$ at the cost of increasing the communication complexity by a factor of roughly $\log k$.

All the constructions described so far are combinatorial in nature and provide probabilistic tracing guarantees. In [17], Boneh and Franklin present an efficient public-key traitor tracing scheme, with deterministic c-tracing based on an algebraic approach. At a high level, the construction can be seen as an extension of the ElGamal encryption scheme to the multi-recipient setting. Rather then being based on the discrete logarithm problem, the scheme of [17] hinges upon the (computationally equivalent) *representation problem*: given $g_1, \ldots, g_{2c}$ elements from a cyclic group of prime order p (called a *base*), as well as $y \doteq g_1^{x_1} \cdot \ldots \cdot g_{2c}^{x_{2c}}$ for random $(x_1, \ldots, x_{2c}) \in \mathbb{F}_p^{2c}$, recover $(x_1, \ldots, x_{2c})$. The public key of the scheme consists of a base $g_1, \ldots, g_{2c}$ and a value y. Encryption is performed by masking the message with y^a for random $a \in \mathbb{F}_p$, and including the values $g_1^a, \ldots, g_{2c}^a$ in the ciphertext, similarly to ElGamal encryption. Decryption requires knowledge of a representation of y with respect to the base $g_1, \ldots, g_{2c}$; given that, one can easily recover the mask y^a from the information in the ciphertext. To reduce user storage to $O(1)$, each user i is given one such representation in an implicit form, consisting of a single value $\theta_i \in \mathbb{F}_p$: a representation of y is then obtained by multiplying θ_i by a codeword $\gamma^{(i)}$ belonging to a fixed and publicly known linear space tracing code Γ.

Boneh and Franklin present several tracing algorithm for their construction, each achieving a different trade-off between efficiency and the assumption on the implementation on the pirate decoder. The less demanding, and hence more widely applicable, tracing model is the *black-box* model, in which the tracing algorithm access the pirate decoder only by querying it on several inputs (this is the same model originally considered in [28]). In fact, the work of [17] refines this model into two variants: *full-access black-box* tracing and *minimal-access black-box* tracing. Full-access black-box tracing requires the pirate decoder to output either the recovered plaintext or the special rejection symbol "invalid" upon each query from the tracing algorithm; minimal-access black-box tracing only demand the illegal box to output either "valid" or "invalid" when fed with a ciphertext from the tracer. To achieve black-box tracing, [17] proceeds by first solving the *black-box confirmation* problem: Given a subset of c users suspected of treachery, the goal is to point out at least one of the traitors whenever the set of suspects actually includes all the traitors, while never accusing an innocent user. A black-box confirmation algorithm implies an exponential-time strategy for tracing, that may have to go through all possible subsets of c users out of a population of N participants. However, the author argue that circumstantial evidence may allow the tracer to narrow the search down to few sets of suspects, in which case black-box confirmation can be used effectively.

To obtain polynomial-time tracing, [17] also considers the *non-black-box* model, based on the assumption that any working pirate decoder must contain a valid representation of y, and that it is possible to efficiently extract such secret information from the pirate box. In this model the authors show how to exploit techniques from Code Theory to efficiently recover *all* the keys that were used to construct the illegal box.

Another scheme achieving black-box traitor tracing appears in [57], where Kiayias and Yung show that if the plaintexts to be distributed can be calibrated to be large (rather than having length independent of the number of traitors and the size of the user population), then it is possible to obtain ciphertexts with constant expansion rate. Their solution is based on the frame-proof codes of Boneh and Shaw [22, 23] which, for an error probability of ϵ, have a required length of $O(c^4 \log(N/\epsilon) \log(1/\epsilon))$. Both plaintexts and ciphertexts in the scheme of [57] have size proportional to the length of the code; the storage complexity is $O(c^4 \log(N/\epsilon) \log(1/\epsilon))$ as well.

Regardless of their black-box versus non-black-box nature, all traitor tracing algorithms described so far are designed to be run by the system manager, and are crucially based on his knowledge of all the secret information that was distributed to the subscribers in the setup phase. As first noticed by Pfitzmann in [69], such design suffers from an intrinsic limitation, namely it cannot provide *non-repudiation.* Indeed, even if the pirate decoder contained all the decryption keys that were used in its construction in an explicit form, and it were possible to extract them from the illegal box, there would always be the possibility that the decoder was in fact purposely constructed by the system manager with the malicious intent of framing innocent users. For this reason, the result of the tracing algorithm in the model of [28] can never be used by the system manager as evidence that the users blamed as pirates are indeed guilty of treachery: since each user shares his decryption keys with the system manager, the mere presence of the user's keys in the pirate box does not imply the user's participation in its construction. To overcome this limitation, [69] introduces the notion of *asymmetric* traitor tracing. In this model, (part of) the secret information that a subscriber gets upon registration is obtained from the system manager in an oblivious fashion. Consequently, the system manager does no longer know all the secret information possessed by the users. Thus, if a pirate decoder is discovered and the tracing algorithm run to expose the identities of the pirates, the system manager obtains some secret data which it could not have produced on its own, so that the result of the tracing algorithm provides actual evidence of the treachery.

The recent work of [27] proposes an orthogonal direction along which the tracing process can be made asymmetric, introducing the notion of *(local) public traceability*: Whereas in traditional traitor tracing schemes only the security manager could execute the tracing procedure (thanks to the knowledge of some piece of information whose secrecy is crucial for the overall security of the system), in a scheme with public traceability everyone can run the tracing algorithm (or at least its preliminary, interactive part which requires the availability of the pirate decoder, in which case one talks of local public traceability). Notice, however, that this *per se* does not guarantee the non-repudiation property of the asymmetric tracing of [69], since in the work of [27] the system manager still knows the secret information of each user.

The work of Kurosawa and Desmedt [60] presents lower bounds for one-time traceability in the traitor tracing model of [28]. They also provide an optimal one-time traceable scheme, which essentially matches their lower bound. The work of [60] also contains a traitor tracing scheme that can be used for multiple transmission (as those in [28]), in the public-key setting. The latter scheme was insecure as presented in [60], but a simple fix was given by Kurosawa and Yoshida in [61], where the authors additionally show how to use linear codes to obtain efficient public-key traitor tracing schemes, which include the construction of [60] and [17] as specific instances. The authors of [60] also consider two constructions for (a variant of) the asymmetric model of [69]. The first scheme guarantees non-repudiation (in a computational sense) by having the secret key of the subscribers being generated by a group of external helper agents. (The system manager can still detect treachery on its own, using the traitor tracing algorithm.) The second scheme works in the same model of [69], and (although efficient) it can only be used in a single transmission.

An efficient traitor tracing scheme providing non-repudiation is due to Kiayias and Yung [56]. Building on previous unsuccessful attempts, the authors show how to exploit *oblivious polynomial evaluation* to efficiently guarantee that the system manager does not get enough information to frame innocent users. The traitor tracing scheme of [56] achieves essentially the same efficiency parameters of [17], in the asymmetric traitor tracing model. In particular, the storage complexity is constant, and communication and encryption complexity are linear in the number of traitors. The scheme is public-key based and provides non-black-box tracing.

An interesting impossibility result in the context of black-box traitor tracing is given in [55], where Kiayias and Yung present an effective *self-protecting* strategy that pirates can employ to immunize their illegal boxes against a large class of traitor tracing algorithm. This mechanism allows the illegal box to detect whether a given ciphertext is valid, *i.e.* contains some content distributed by the data supplier, or invalid, *i.e.* it was supplied by the tracer to expose the traitors identities. The strategy works assuming a coalition of $\omega(\log N)$ traitors and is effective in evading tracing in schemes that strive to achieve low communication complexity, including the efficient ones proposed in [60] and [17]. Remarkably, the original constructions of [28, 29] are not affected by this piracy mechanism, so that the work of [55] in a sense provides a separation between the black-box capabilities of [28, 29] and of [60] and [17].

We remark that most traitor tracing schemes in the literature do not aim at guaranteeing the traceability of traitors when the *entire* user population conspires against the data provider to create pirate decoders. If one insists on such strong guarantee (called *full-traceability*), the current best solution, due to Boneh *et al.* [21], entails $O(\sqrt{n})$ communication encryption and public storage complexity, and constant private storage.

3.2.2 Tracing Based on Leaked Decrypted Content

Another natural piracy strategy is the following. The pirates first subscribe to the system, and whenever scrambled content is distributed by the data supplier, they decrypt it and then rebroadcast it in the clear to their own set of customers. It is arguably less practical for a group of pirates to follow this strategy, as it involves the maintenance of a separate broadcast infrastructure to redistribute the content to the black-market. This clearly entails higher cost for the pirates, and also higher visibility and consequentially higher risk of being discovered. For these reasons, fewer proposals of traitor tracing schemes actually address this kind of piracy threat.

One such proposal is due to Fiat and Tassa [44] who employ watermarking techniques (similar to Boneh and Shaw's frame-proof codes [22]) to allow the system manager to trace the identities of the traitors in case that a pirate rebroadcast is discovered. The model assumes that the data supplier has an unlimited stream of content to broadcast, split into equal-length *segments*. Each segment is delivered to the user population via a dedicated *transmission*. In fact, to achieve c-traceability, watermarking is used to produce several *variants* of each segment. Variants are distributed in a two-step transmission: first, personalized keys are distributed to the users via any broadcast encryption scheme; then, each variant is encrypted under its own key.

Tracing in this model is an iterative process: once a pirate rebroadcast has been found, the tracer start to observe which variant of the segment was available to the pirates, and bases the broadcast of the subsequent segments on such observation. The efficiency of dynamic traitor tracing schemes is measured in terms of the number of distinct variants needed, and of the number of rounds needed in order to nail down the identities of all the traitors (*convergence time*).

In [44], the authors present three deterministic schemes. The first scheme requires no more than $c+1$ variants for each segment, but its convergence time is exponential in N. The second scheme requires $2c+1$ variants per segment, but detects all the traitors within $O(c \log N)$ steps. The third scheme requires $c+1$ variants, but its convergence time is $O(3^c c \log N)$.

Berkman *et al.* [9] improved the results of [44] providing a scheme that, using $c+1$ variants, can trace c traitors in $\Theta(c^2 + c \log N)$ rounds. This scheme is quite involved and the hidden constants are large. Alternatively, the authors define other two schemes: the first locates c traitors in $O(c^2/d + \log N)$ rounds, using $c+d+1$ variants ($1 \leq d \leq c$); the second requires $cd+1$ variants ($d \geq 2$) and locates c traitors in $O(c \log_c N)$ rounds. The work of [9] also contains lower-bound showing that the above results are optimal for almost any value of d. In particular, the authors prove that, using $c+d$ variants, the converge time to locate c traitors (when c is not *a priori* known) is $\Omega(c^2/d + c \log_{d+1} N)$. This bound is tight when d is a constant or when it is super-linear in c; otherwise the upper-bound is off by a factor of $\log c$.

Safavi-Naini and Wang [71, 72] points out two limitations in the model of [44]. First, dynamic traitor tracing is subject to the *delayed rebroadcast* attack. The attack is effective because, during tracing, a dynamic scheme encodes a segment s based on the feedback observed on the pirate rebroadcast for segment $s-1$. Second, dynamic traitor tracing schemes impose a substantial computational overhead on the system manager, who is required to compute the assignment of variants to users on the fly. To address these shortcomings, Safavi-Naini and Wang [71] propose the notion of *sequential traitor tracing*. In order to avoid the dependence of the broadcast on the feedback from the pirate rebroadcast (which is the cause of both the above limitations in dynamic traitor tracing), variant-assignment for each segment is based on a predefined table. The work of [72] presents two concrete schemes: one based on a number-theoretic construction, and another based on the use of error-correcting codes.

3.3 Trace and Revoke Schemes

In Section 3.1 and Section 3.2, we cover the literature related to broadcast encryption schemes and traitor tracing schemes. It is interesting to notice that broadcast encryption and traitor tracing schemes address orthogonal issues: the latter allows to discover the identities of misbehaving users that can then be excluded from subsequent broadcasts with the revocation capabilities of a broadcast encryption scheme. Such observation, first explicitly expressed in [45], has led most of the subsequent related literature to treat broadcast encryption and traitor tracing jointly, under the name of *Trace and Revoke schemes*.

3.3.1 Secret-Key Trace and Revoke Schemes

The work of Stinson and Wei [78] presents a very general (though not efficient) technique to add broadcasting capabilities to a traitor tracing scheme by expanding each secret key in the universe of keys into a set of keys.

This approach was then extended by Gafni *et al.* in [45], where the authors discuss methods to augment broadcast encryption schemes with traceability functionalities, and conversely, to build revocation capabilities into traitor tracing schemes. In the first case, the idea is to introduce a suitable number of clones of the universe of keys, and for each key that a user was supposed to get in the original scheme, he now receives a random key-clone from the corresponding set. Tracing is then based on the fact that different users are not likely to share the same random key-clones. If the storage complexity in the original broadcast encryption scheme is $\kappa_s > 4c^2 \log N$, then the storage complexity of the resulting scheme remains unchanged, while the communication complexity deteriorates by a factor of $2c^2$, where c measures the level of security of the traceability algorithm (*c-traceability*, see Section 3.2).

In the second case (adding revocation to traitor tracing schemes), the universe of keys is again cloned, but this time each user gets all the key-clones corresponding to his original set of keys, so that the storage per user increases. Then, the resulting storage and communication complexity increase by a factor of r/c.

In [65], Naor *et al.* present a symmetric-key trace and revoke scheme for the *stateless receiver* setting. In this setting, each user is given a fixed set of keys which cannot be updated throughout the lifetime of the system. In particular, keys do not change when other users join or leave the system, nor they evolve based on the history of past transmissions. Instead, each transmission must be decrypted solely on the base of the fixed initial configuration of each user's decryption device. The stateless receivers setting realistically models common practical scenarios, in which receivers might not be constantly on-line to view past history or update their secret keys, or in which keys might be put "once-and-for-all" into a write-once tamper-resistant device.

An important contribution of [65] is to refine the class of OR protocols [62] by defining the *Subset-Cover Framework*, a powerful paradigm for the formulation and security analysis of revocation schemes. The main idea of the framework is to define a family $\mathcal{S}$ of subsets of the universe $\mathcal{N}$ of users in the system, and to associate each subset with a key, which is somehow made available to all the users belonging to the given subset. When the sender wants to broadcast a session key to all the subscribers but those in some set $\mathcal{R}$, it "covers" the set of privileged users using subsets from the family $\mathcal{S}$ (*i.e.* the sender determines a partition of $\mathcal{N} \setminus \mathcal{R}$, where all the subsets are elements of $\mathcal{S}$), and then encrypts the session key with all the keys associated to the subsets in the found partition.

A revocation scheme within the Subset-Cover framework is fully specified defining the particular Subset-Cover family $\mathcal{S}$ used, the algorithm to find the cover for the authorized set of subscribers and the key assignment employed to (implicitly) deliver to each user the key corresponding to the sets to which he belongs. Specific examples include the *Complete Subtree* (CS) method and the *Subset Difference* (SD) method: the first is based on an information-theoretic approach and achieves storage complexity $O(\log N)$ and communication complexity $O(r \log N)$; the second employs computational techniques to lower the communication complexity to $O(r)$ at the cost of increasing the storage complexity by an extra $\log N$ factor. Additionally, both schemes support an unbounded number of broadcasts, and allow to revoke an *a priori* unbounded number of users. In particular, even the coalition of *all* the "non-privileged" users combined cannot decrypt a given transmission, even if this set is adaptively chosen by a central adversary. Finally, the above features also imply that consecutive broadcasts can revoke arbitrary and potentially unrelated subsets of users, and no "key maintenance" is necessary.

In the CS scheme, users are organized in a tree structure as leaves of a complete binary tree $\mathcal{T}_{CS}$ of height $\log N$. The Subset-Cover family $\mathcal{S}_{CS}$ is then set to be the collection of all the complete subtrees of $\mathcal{T}_{CS}$. More precisely, if v_j is a node in $\mathcal{T}_{CS}$, the generic $S_j \in \mathcal{S}_{CS}$ is the set of all the leaves of the complete subtree of $\mathcal{T}_{CS}$ rooted at v_j (thus, in this case $|\mathcal{S}_{CS}| = 2N - 1$). To associate a key to each element of $\mathcal{S}_{CS}$, the trusted initializer of the system assigns a random number $\mathcal{L}_j$ to each node v_j in $\mathcal{T}_{CS}$, which is then used to perform all the encryption/decryption operations relative to the subset S_j. Furthermore, since each user needs to know the keys corresponding to all the subsets he belongs to, during the subscription step, each subscriber get the keys relative to each node in the path from the root down to the leaf representing the subscriber.

To improve the communication complexity of the CS method, in the SD scheme each user belongs to more subsets, thus allowing for greater freedom (and hence higher efficiency) in the choice of the cover. As before, users are associated to the leaves of a complete binary tree $\mathcal{T}_{SD}$, but the generic subset S_{ij} is now defined in term of two nodes $v_i, v_j \in \mathcal{T}_{SD}$ (with v_i ancestor of v_j), called respectively *primary root* and *secondary root* of S_{ij}. Specifically, each subset S_{ij} consists of all the leaves of the subtree rooted at v_i except those in the subtree rooted at v_j. The initialization phase follows a computational approach that defines the keys $\mathcal{L}_{ij}$'s in term of a limited amount of random information, called *labels*. To do so, the system manager associates a random number $\textsc{Label}_i$ to each internal node v_i in $\mathcal{T}_{SD}$. Then, to generate a label for each subset S_{ij}, a pseudo-random number generator [13] $\mathcal{G} : \{0,1\}^m \longrightarrow \{0,1\}^{3m}$ is used, where m is the desired length of the keys $\mathcal{L}_{ij}$. For notational convenience, given an input x, we denote with $\mathcal{G}_L(x)$ the m leftmost bits of $\mathcal{G}(x)$, with $\mathcal{G}_R(x)$ the m rightmost bits of $\mathcal{G}(x)$, and with $\mathcal{G}_M(x)$ the remaining m central bits of $\mathcal{G}(x)$. Using the generator $\mathcal{G}$, we can express the relationship between $\textsc{Label}_{ij}$ and $\textsc{Label}_{il}$ (with v_l parent of v_j) as follows:

$$\textsc{Label}_{ij} = \begin{cases} \mathcal{G}_L(\textsc{Label}_{il}), & \text{if } v_j \text{ is the left child of } v_l, \\ \mathcal{G}_R(\textsc{Label}_{il}), & \text{if } v_j \text{ is the right child of } v_l. \end{cases}$$

Finally, the key $\mathcal{L}_{ij}$ associated to the subset S_{ij} can be derived from $\textsc{Label}_{ij}$ as $\mathcal{L}_{ij} = \mathcal{G}_M(\textsc{Label}_{ij})$. Under this key-assignment, a user u can derive the $O(N)$ keys corresponding to subsets he belongs to, in terms of the $O(\log^2 N)$ labels $\{\textsc{Label}_{ij}\}$ where v_i is an ancestor of u and v_j is the sibling of a lower-level ancestor of u.

In [65], Naor *et al.* present a black-box tracing algorithm that works in the threshold traitor tracing model. Such method can be applied to any revocation scheme in the Subset-Cover framework satisfying the property that, for any subset $S \in \mathcal{S}$, there exist two roughly equal-sized subsets S_1, S_2 that partition S (*bifurcation property*). The tracing guarantees provided by the scheme of [65] are slightly relaxed with respect to standard black-box traitor tracing: the goal of the tracer is to *either* discover the identity of one of the traitors *or* determine a partition of $\mathcal{N} \setminus \mathcal{R}$ that render the illegal decoder useless. At a high level, the idea is to start with a partition of $\mathcal{N} \setminus \mathcal{R}$ and iteratively split one of the subset containing at least one traitor until we either get a singleton (which exposes the identity of a traitor), or we obtain a partition which renders the illegal box useless. To implement this strategy, the authors additionally present a *subset tracing* subroutine that efficiently tests whether a given subset contains at least one traitor using a binary-search-like approach.

An improvement on the SD method is due to Halevy and Shamir [51], under the name of *Layered Subset Difference* (LSD) method. The construction is again based on the idea on organizing the users as leaves of a complete binary tree, but in [51] the authors additionally structure the levels of the tree into *layers*, and introduce *shortcuts* to jump from one level to another in a lower layer. This allows to reduce the storage complexity to $O(\log^{1+\epsilon} N)$, for any $\epsilon > 0$ at the cost of increasing the communication complexity by a constant factor of $1/\epsilon$.

More in details, the LSD scheme reduces the size of the family $\mathcal{S}_{SD}$ by only considering a subcollection $\mathcal{S}_{LSD}$ of *useful* subsets. The key observation to reach this goal is that any subtree difference set S_{ij} can be rewritten as the disjoint union $S_{ik} \cup S_{kj}$, for any node v_k lying in the path from v_i to v_j.

To define the sub-collection $\mathcal{S}_{LSD}$, consecutive levels of the tree $\mathcal{T}_{SD}$ are grouped into *layers*, which induces the notions of *local* and *special* subsets of $\mathcal{S}_{SD}$. In particular, local subsets are those whose primary and secondary roots both lie within the same layer, while *special* subset are those having as their primary root a node lying exactly on the boundary between two adjacent layers. The sub-collection $\mathcal{S}_{LSD}$ consists exactly of all the local and special subsets of $\mathcal{S}_{SD}$. In this way, the number of proto-keys that each user needs in order to decrypt each broadcast can be reduced, while the Center can preserve the functionalities of the system by at most doubling the size of the cover. This is because any subset $S_{ij} \in \mathcal{S}_{SD}$ can be obtained as the union of a local subset and a special subset in $\mathcal{S}_{LSD}$.

In [51], the authors also considered an alternative approach to the problem of specifying the set of revoked users $\mathcal{R}$ that should not be a able to recover the broadcasted message. Such technique is based on the use of *Inclusion-Exclusion Trees* (IE-Trees), which offer a convenient way of describing a large set of privileged users with relative few nested inclusion and exclusion conditions on the nodes of the tree $\mathcal{T}_{SD}$.

The advantage of such technique is that from an IE-Tree it is possible to derive a cover whose size is proportional to the number of conditions specified by the IE-Tree itself.

In applications where user storage is at premium (*e.g.,* in the context of sensor networks) requiring $\omega(\log N)$ storage at the end user could be problematic. In [49], the authors argue that, in such scenarios, it would be preferable to relax the *stateless receivers* requirement and assume that the receivers are equipped with a smaller read-write memory to store the user's secret information. Moving from these considerations, in [49] the authors propose a new broadcast encryption scheme, called *Stratified* Subset Difference (SSD) method, which maintains the $O(r)$ communication complexity of the SD method, but requires only $O(\log N)$ storage complexity. The computation complexity, however, becomes proportional to $O(N^{1/d})$, where d is a parameter of the scheme.

The recent paper of [54] presents three new broadcast encryption schemes for stateless receivers that fit within the Subset Cover framework of [65]. The first scheme, the *Skipping* scheme, improves the transmission overhead when the fraction of revoked users (r/N) is above a suitable threshold, but performs poorly below it. The degree of the improvement and the exact value of this threshold can be controlled by two parameters, p and c: the larger p, the larger the improvement; the larger c, the lower the threshold. However, the storage requirement for the Skipping scheme grows with p and c as c^{p+1}. The second scheme, the *Cascade* scheme, is developed to address settings with small ratio of revoked users, and results in a solution that has the same transmission overhead as the SD scheme below the threshold, and slightly better above it. The storage per user is roughly $c \log^2 N$, where the parameter c controls the threshold as in the Skipping scheme. The third scheme proposes a combination of the above two schemes, yielding a transmission overhead which is never worse than that of SD and is actually better for value of r above the threshold. The storage requirement is roughly the sum of the storage requirements of the previous two schemes.

The work of [53] is driven by the assumption that most existing solutions have overemphasized the relative importance of the communication complexity with respect to the other parameters of the scheme. The authors then describe a generic transformation that reduces the storage complexity and/or decryption complexity of a broadcast encryption scheme at the expenses of its communication and encryption complexity. The main idea of the transformation is to arrange the user population in a high-degree (rather than binary) hierarchy, and iteratively apply the given scheme to small subset of users. Although this typically results in a worsening (rather than an improving) of the asymptotic performance of the original scheme, the "transformed" version may have better concrete performance for some choice of the parameters.

3.3.2 Public-Key Trace and Revoke Schemes

The work surveyed so far focused on the *secret-key* setting, where the only entity able to broadcast to the user population is the trusted initializer and maintainer of the system. A *public-key* solution, instead, allows the trusted system manager to publish a short public key that enables any untrusted party to selects its own set of privileged users and broadcast to them using the underlying infrastructure.

Two trace and revoke schemes have been proposed in the public-key setting, by Naor and Pinkas in [67] and Tzeng and Tzeng in [80]. Both schemes have similar constructions and efficiency parameters: the common idea is to combine Berkovits' proposal [10] with Feldman's verifiable secret sharing [42], by lifting the shares constituting the broadcast message to the exponent. This allows (under standard computational assumptions) to reuse the same polynomial across broadcasts, and thus to reduce the communication complexity from $O(N - r)$ to $O(r)$. Below, we describe the construction of [80], who emphasize more the public-key nature of their scheme. To achieve security against up to r revoked users, the sender sets up a public key of size $O(r)$ by selecting a random

r-degree polynomial P over a finite field $\mathbb{F}_p$ and a generator g of a cyclic group $\mathbb{G}$ of order p, and then publishing:

$$\langle g, g^{P(0)}, g^{P(1)}, \ldots, g^{P(r)} \rangle.$$

Upon joining the system, a user i gets the value of the polynomial P at a random point x_i. In a public-key system, any party can broadcast a secret $s \in \mathbb{G}$ to the users in the system, possibly revoking up to r of them. To do so, the broadcaster selects a random $d \in \mathbb{F}_p$ and broadcast the message:

$$\langle sg^{dP(0)}, g^d, (z_1, g^{dP(z_1)}), (z_2, g^{dP(z_2)}), \ldots, (z_r, g^{dP(z_r)}) \rangle$$

where $z_1, z_2, \ldots, z_r$ are the identities of the revoked users. Message decryption amounts to computing the value $g^{dP(0)}$, since then it is possible to recover s from the first component of the ciphertext. To compute $g^{dP(0)}$, user i first raises g^d (included in the ciphertext) to his secret share $P(x_i)$, and then uses interpolation (in the exponent) of the resulting value together with the r values in the broadcast. Notice that any revoked user is prevented from recovering the message, since the value resulting from his share is already included in the broadcast, so that interpolation is not possible. The storage complexity of the scheme is $O(1)$, while the communication complexity is linear in r.

Both trace and revoke schemes proposed in [67, 80] provide traceability guarantees and efficiency comparable to the traitor tracing scheme of [17]. They achieve a clean combination of the ability to revoke up to an *a priori* bounded number of users, with black-box confirmation guarantees. The also provide non-black-box traitor tracing, assuming the key extracted from the pirate decoder is in a suitable *canonical form.*

Chapter 4

Public-Key Broadcast Encryption for Stateless Receivers

4.1 Introduction

As discussed in Chapter 3 (Section 3.1), one important distinction between various broadcast encryption schemes is whether they are public-key or symmetric-key based. In the latter variant, only the trusted designer of the system can broadcast data to the receivers. In other words, in order to encrypt the content, one needs to know some sensitive information (typically, the secret keys of all the users of the system) whose disclosure will compromise the security of the system itself. Even though symmetric key broadcast encryption is sufficient for many applications, it has a few shortcomings. For example, it requires the sender to store all the secret keys of the system, making it a single point of failure. Additionally, in certain situations we would like to allow possibly untrusted users to broadcast information, which is not possible in the symmetric setting.

In contrast, in the public-key setting the trusted designer of the system publishes a short public key which enables anybody to broadcast data, thus overcoming the above mentioned deficiencies of the symmetric setting.

In this chapter, based on work first published as [34], we describe a construction of public-key broadcast encryption for the *stateless receivers* setting, in which users hold decryption keys that cannot be updated as the system evolves (*cf.* also Chapter 3, Section 3.3.1).

Our construction extends the *Subset-Cover Framework* (*cf.* Section 3.3.1), initially proposed by Naor et al. [65] for the case of symmetric-key broadcast encryption, to the more challenging public-key setting. In fact, in [65] Naor et al. briefly mention that the general Subset-Cover framework can in principle be adapted to the public-key setting, by having the secret key of each subset S_i replaced by some pair of public/secret keys (PK_i, SK_i). Unfortunately, this creates the problem of how to compress the exponentially-many public keys $PK_1, \ldots, PK_w$ to a manageable size. Naor *et al.* [65] hint that the tools from Identity-Based Cryptography (*cf.* Chapter 2 Section 2.4) could help for the (inefficient) CS method. However, Identity-Based Encryption (IBE), (*cf.* Chapter 2, Section 2.4.1) alone is not sufficient for the more efficient SD method, since it cannot support the key compression techniques of [65]. In fact, suitable key compression methods are much harder to come by in the public-key setting, and the question of efficiently extending the SD method to the public-key setting was left as an interesting open problem in [65].

4.2 Our Results

We resolve this problem in the affirmative, leveraging the concept of *Hierarchical Identity-Based Encryption* (HIBE) (*cf.* Chapter 2, Section 2.4.2). In particular, we show that one can get essentially all the benefits of the symmetric key versions of the SD/LSD methods (including the same small storage per user) in the public-key setting, while having a fixed *constant size* public key. As an intermediate step toward this goal, we indicate which changes should be made to the general Subset-Cover framework of [65] in order to translate it to the public-key setting, and also formally verify that "plain" IBE is indeed sufficient to translate the (inefficient) CS method to the public-key setting. The particular parameters we get can be summarized as follows when revoking r out of

N total users (in all cases, the public key size and the storage of the Center are $O(1)$):

- **CS method**. The ciphertext consists of $r \log(N/r)$ identity based encryptions, each users stores $O(\log N)$ keys and needs to perform a single identity based decryption.
- **SD method**. The ciphertext consists of $(2r - 1)$ hierarchical identity based encryptions (of "depth" at most $\log N$ each), each users stores $O(\log^2 N)$ keys and needs to perform a single hierarchical identity based decryption.
- **LSD method**. For any $\epsilon > 0$, the ciphertext consists of $O(r/\epsilon)$ hierarchical identity based encryptions (of "depth" at most $\log N$ each), each users stores $O(\log^{1+\epsilon} N)$ keys and needs to perform a single hierarchical identity based decryption.

Comparison to Existing Public-Key Schemes. There already exist several (quite similar to each other) public-key broadcast encryption schemes [67, 80, 36] in the stateless receivers scenario, all based on the decisional Diffie-Hellman (DDH) assumption. However, all these schemes can revoke up to an a-priori fixed number of users, $r_{\max}$. Moreover, the size of the transmission is $O(r_{\max})$ even if no users are revoked. In contrast, the SD/LSD methods allow to revoke a dynamically changing (and potentially unbounded) number of users r, at the cost of having $O(r)$-size ciphertext transmission. More importantly, the reason the schemes of [67, 80, 36] support only a bounded number of revoked users, is that the public key (as well as encryption/decryption times) are proportional to $r_{\max}$. In contrast, the analogs of CS/SD/LSD schemes we construct all have a constant size public key, and the decryption time is at most logarithmic in the total number of users N. Finally, the schemes of [67, 80, 36] support only a limited form of traitor tracing (either "non-black-box" or "black-box confirmation"), while (as was shown in [65]) the CS/SD/LSD methods enjoy a significantly more powerful kind of "black-box" traitor tracing.

4.3 Formal Model

4.3.1 BE: Syntax

Definition 27 (Public-Key Broadcast Encryption Scheme).

A BE scheme $\mathcal{E}_{\mathsf{BE}}$ is a four-tuple of probabilistic polynomial-time algorithms (Setup, $\mathsf{Register}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$), where:

- Setup, the *key generation algorithm*, is run by the Center to set up the parameters of the scheme. Setup takes as input a security parameter 1^λ and possibly $1^{r_{\max}}$ (where $r_{\max}$ is a revocation threshold, *i.e.* the maximum number of users that can be revoked). The input also includes the total number N of users in the system. Setup generates the public key $\mathsf{params}_{\mathsf{BE}}$ and the master secret key $\mathsf{master}_{\mathsf{BE}}$. The Center publishes $\mathsf{params}_{\mathsf{BE}}$ and keeps $\mathsf{master}_{\mathsf{BE}}$ secret.

- $\mathsf{Register}$, the *registration algorithm*, is run by the Center to compute the secret initialization data for a new user. $\mathsf{Register}$ takes as input the master secret key $\mathsf{master}_{\mathsf{BE}}$ and the identity u of the user, and outputs the new secret key SK_u.

- $\mathsf{Encrypt}$, the *encryption algorithm*, is used to encapsulate a given session key s within an enabling block $\mathcal{B}$. $\mathsf{Encrypt}$ takes as input the public key $\mathsf{params}_{\mathsf{BE}}$, the session key s and a set $\mathcal{R}$ of revoked users (with $|\mathcal{R}| \le r_{\max}$, if a threshold has been specified to the Setup algorithm) and returns the enabling block $\mathcal{B}$ to be broadcast.

- $\mathsf{Decrypt}$, the *decryption algorithm*, takes as input the public key $\mathsf{params}_{\mathsf{BE}}$, the identity u of a user, the user's secret key SK_u and an enabling block $\mathcal{B}$. $\mathsf{Decrypt}$ returns a session key s or the special rejection symbol $\perp$.

A BE scheme $\mathcal{E}_{\mathsf{BE}}$ should satisfy the following correctness constraint: for any pair $(\mathsf{params}_{\mathsf{BE}}, \mathsf{params}_{\mathsf{BE}})$ output by $\mathsf{BE.Setup}(1^\lambda, 1^{r_{\max}}, N)$, any $\mathcal{R} \subseteq \mathcal{N}, (|\mathcal{R}| \le r_{\max})$, any user $u \in \mathcal{N} \setminus \mathcal{R}$ with secret key SK_u (properly generated for user u) and any session key s:

$$s = \mathsf{BE.Decrypt}(\mathsf{params}_{\mathsf{BE}}, u, \mathsf{SK}_u, \mathsf{BE.Encrypt}(\mathsf{params}_{\mathsf{BE}}, \mathcal{R}, s)).$$

4.3.2 BE: Security

A BE scheme $\mathcal{E}_{\mathsf{BE}}$ is secure against chosen-ciphertext attack if for all polynomially-bounded N, no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage in the following game:

Setup: The challenger runs algorithm $\mathsf{BE.Setup}(1^\lambda, 1^{r_{\max}}, N)$; it then gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{BE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{BE}}$ secret.

Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. $\textsc{Corrupt}(u)$: the challenger runs algorithm $\mathsf{BE.Register}(\mathsf{master}_{\mathsf{BE}}, u)$ to generate the private key SK_u corresponding to user u, and sends SK_u to $\mathcal{A}$. Notice that if a bound r_{max} was specified in Setup, then the adversary is restricted to corrupt at most r_{max} distinct users via $\textsc{Corrupt}(\cdot)$ queries.

2. $\textsc{Decrypt}(u, \mathcal{B}_j)$: the challenger runs algorithm $\mathsf{BE.Register}(\mathsf{master}_{\mathsf{BE}}, u)$ to recover the private key SK_u corresponding to user u. It then runs algorithm $\mathsf{BE.Decrypt}(\mathsf{params}_{\mathsf{BE}}$, u, SK_u, $\mathcal{B}_j)$ and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs two equal length session keys $s_0, s_1 \in \mathcal{M}$ on which it wishes to be challenged. The challenger picks a random bit $b \in \{0,1\}$, sets $\mathcal{B}^* \doteq \mathsf{BE.Encrypt}(\mathsf{params}_{\mathsf{BE}}, \mathcal{R}^*, s_b)$, where $\mathcal{R}^* \doteq \{u \mid \mathcal{A}$ asked a query $\textsc{Corrupt}(u)\}$. The challenger sends $\mathcal{B}^*$ as a challenge to $\mathcal{A}$.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is of the form:

1. $\textsc{Decrypt}(u, \mathcal{B}_j)$: the challenger first checks that either $u \in \mathcal{R}^*$ or $\mathcal{B}_j \neq \mathcal{B}^*$ and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0,1\}$ and wins the game if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as an BE.IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the BE scheme $\mathcal{E}_{\mathsf{BE}}$ as:

$$\mathsf{Adv}_{\mathsf{BE},\mathcal{A}} = \left| \Pr[b = b^*] - \frac{1}{2} \right|.$$

The probability is over the random bits of the challenger and of the adversary.

We can also define the notion of selective-identity chosen-ciphertext security for BE (BE.IND-sID-CCA). The game is exactly as for the BE.IND-ID-CCA case, except that the adversary $\mathcal{A}$ discloses to the challenger the set $\mathcal{R}^*$ of revoked users *before* the **Setup** phase.

Definition 28 (Chosen-Ciphertext Security for BE).
A BE scheme $\mathcal{E}_{\mathsf{BE}}$ is $(\tau, q_{\mathrm{ID}}, q_C, \varepsilon_{\mathsf{BE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure if for any τ-time BE.IND-ID-CCA (resp. BE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen Corrupt queries and at most q_C chosen Decrypt queries, it holds that $\mathsf{Adv}_{\mathsf{BE},\mathcal{A}} < \varepsilon_{\mathsf{BE}}$.

We define chosen-plaintext security for a BE scheme according to an attack game that is exactly as in the IND-ID-CCA game, except that the adversary is not allowed to issue Decrypt queries. The adversary may still issue adaptive chosen Corrupt queries.

Definition 29 (Chosen-Plaintext Security for BE).
A BE scheme $\mathcal{E}_{\mathsf{BE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{BE}})$-IND-ID-CPA (resp. IND-sID-CPA)-secure if it is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{BE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure.

A Generalization of the Notion of Chosen-Ciphertext Security. In the definition of an IND-ID-CCA game, the adversary is allowed to "play" with the decryption machinery as she wishes, subject only to the condition that she does not ask about enabling blocks "closely related" to her challenge $\mathcal{B}^*$.

In formalizing the notion of "close relationship," the usual treatment is to impose a minimal restriction to the adversary, just disallowing her to submit the challenge itself to the decryption machinery. As already noted in [75, 2], such a mild constraint does in turn restrict too much the class of schemes that can be proven secure, excluding even schemes that ought to be considered secure under a more intuitive notion. For this reason, in some scenarios, it seems more reasonable to consider a variant of the IND-ID-

CCA-security, to which we will refer to as *generalized* chosen-ciphertext (IND-ID-gCCA) security, following the terminology introduced in [2].

In a generalized chosen-ciphertext game, the set of enabling blocks the adversary is forbidden to ask about is defined in term of an efficiently computable equivalence relation $\Re(\cdot,\cdot)$. In fact, in the case of a broadcast (as opposed to ordinary) encryption, there is no unique decryption machinery, since the decryption algorithm can be used with the secret key of any legitimate user. For this reason, in our setting we need to consider a *family* of efficient equivalence relations $\{\Re_u(\cdot,\cdot)\}$, one for each user u. As in the regular case [2], the equivalence relation $\Re_u(\cdot,\cdot)$ corresponding to each user u needs to be *u-decryption-respecting*: equivalent enabling blocks under $\Re_u(\cdot,\cdot)$ are guaranteed to have exactly the same decryption according to the secret data of user u. Finally, this family is an *explicit* parameter of the scheme, and an IND-ID-gCCA proof of security is always relative to the specified decryption-respecting family $\{\Re_u(\cdot,\cdot)\}$.

Given a decryption-respecting family $\{\Re_u(\cdot,\cdot)\}$, we define generalized chosen-ciphertext security for a BE scheme by considering an attack game that is exactly as in the IND-ID-CCA case, except that we disallow the adversary to issue DECRYPT queries for which $\Re_u(\mathcal{B},\mathcal{B}^*)$ holds.[1] Notice that the adversary may still issue adaptive chosen CORRUPT queries.

Definition 30 (GENERALIZED CHOSEN-CIPHERTEXT SECURITY FOR BE)**.**
A BE scheme $\mathcal{E}_{\mathsf{BE}}$ is $(\tau, q_{\mathrm{ID}}, q_C\varepsilon_{\mathsf{BE}})$-IND-ID-$\{\Re_u(\cdot,\cdot)\}$-gCCA (resp. IND-sID-$\{\Re_u(\cdot,\cdot)\}$-gCCA)-secure if for any τ-time BE.IND-ID-CCA (resp. BE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen CORRUPT queries and at most q_C chosen DECRYPT queries (parameterized by the decryption-respecting family $\{\Re_u(\cdot,\cdot)\}$), it holds that $\mathsf{Adv}_{\mathsf{BE},\mathcal{A}} < \varepsilon_{\mathsf{BE}}$.

[1]This definition encompasses the notion of IND-ID-CCA security as the special case where each relation $\Re_u(\cdot,\cdot)$ is the equality relation.

4.4 Public-Key Extension of the CS Method

As already mentioned in Section 4.2, a naïve approach to the problem of transposing the CS method to the asymmetric setting yields a total number of $2N - 1$ public keys, which makes it unfeasible to maintain them all explicitly. To overcome this problem, we need a scheme that allows for an implicit and compact representation of all the public keys, from which to easily extract the needed information: the functionalities of any Identity-Based Encryption scheme come handy in this situation, yielding the efficient solution described below.

As a preliminary step, a fixed mapping is introduced to assign an identifier $\mathrm{ID}(S_j)$ to each subset S_j of the family $\mathcal{S}_{CS}$. For example, a simple mapping could consist of labeling each edge in the complete binary tree $\mathcal{T}_{CS}$ with 0 or 1 (depending on whether the edge connects the node with its right or left child), and then assign to the subset S_j rooted at v_j the bitstring obtained reading off all the labels in the path from the root down to v_j.

Afterwards, the Center runs the Setup algorithm of an IBE scheme to create an instance of the system in which it will play the role of the Root. Then, the Center publishes the parameters $\mathsf{params}_{\mathsf{IBE}}$ of the IBE system and the description of the mapping used to assign an identifier to each subset: these two pieces of data constitute the compact representation of the public keys of all the subsets in the cover family $\mathcal{S}_{CS}$ for the CS method, and require only $O(1)$ space.

To generate the private key $\mathcal{L}_j^{\mathrm{PRI}}$ associated to a subset $S_j \in \mathcal{S}_{CS}$, the Center sets:

$$\mathcal{L}_j^{\mathrm{PRI}} \leftarrow \mathsf{IBE.Extract}(\mathsf{params}_{\mathsf{IBE}}, \mathrm{ID}(S_j), \mathsf{master}_{\mathsf{IBE}}).$$

At this point, the Center can distribute to each subscriber the private data necessary to decrypt the broadcast, as in the original, symmetric scheme. Moreover, whenever a (not necessarily trusted) party wants to broadcast a message, it can encrypt the session key s used to protect the broadcast under the public keys $\mathcal{L}_{i_j}^{\mathrm{PUB}} = \mathrm{ID}(S_{i_j})$ relative to all the subsets that make up the cover of the chosen set of privileged users. To this aim, this party only needs to know the parameters of the IBE system $\mathsf{params}_{\mathsf{IBE}}$ and the description of the mapping $\mathrm{ID}(\cdot)$, and then it can compute:

$$B_j \leftarrow \mathsf{IBE.Encrypt}(\mathsf{params}_{\mathsf{IBE}}, \mathrm{ID}(S_{i_j}), s)$$

for all the subset S_{i_j} in the cover.

Security. The IND-ID-CCA1-security[2] of the scheme follows almost immediately from the IND-ID-CCA security of IBE. Indeed, when revoking some set $\mathcal{R}$ of users, the adversary does not learn any of the secret keys used for transmitting the message to the remaining users $\mathcal{N}\backslash\mathcal{R}$ (since only sets disjoint from $\mathcal{R}$ are used in the cover), so the IND-ID-CCA1-security of broadcast encryption follows by a simple hybrid argument over the sets covering $\mathcal{N}\backslash\mathcal{R}$.

A Concrete Instantiation. If we apply the above idea in conjunction with the specific IBE scheme proposed in [81], the public-key extension matches the original variant in all the efficiency parameters; more precisely, the storage requirement on each user is still $O(\log N)$ and the transmission rate is $r \log \frac{N}{r}$, where $r = |\mathcal{R}|$.

4.5 Public-Key Extension of the SD Method

To improve the transmission rate, the SD scheme uses a more sophisticated Subset-Cover family $\mathcal{S}$: each user will belong to more subsets, thus allowing for greater freedom (and hence higher efficiency) in the choice of the cover. On the flip side, this will create a problem of compressing the user's storage which will need to be addressed.

Due to the large number of subsets that contain a given user, it is no longer possible to employ an information-theoretic key assignment, directly associating a random key to each element in the family $\mathcal{S}$ (as it was done in the CS method), because this would require each subscriber to store a huge amount of secret data: to overcome this problem, a more involved, computational technique is required.

[2] IND-ID-CCA1-security is different from IND-ID-CCA-security for the fact that in the case of IND-ID-CCA1 the adversary is disallowed to ask DECRYPT queries after she received the challenge ciphertext.

The idea behind the solution proposed in [65] is to derive the set of actual keys $\{\mathcal{L}_{ij}\}$ from some set of "proto-keys" $\{\mathcal{P}_{ij}\}$ satisfying the following properties:

1. given the proto-key $\mathcal{P}_{ij}$ it is easy to derive the key $\mathcal{L}_{ij}$;
2. given the proto-key $\mathcal{P}_{il}$ it is easy to derive the proto-key $\mathcal{P}_{ij}$, for any node v_j descendent of node v_l;
3. it is computationally difficult to obtain any information about a proto-key $\mathcal{P}_{ij}$ without knowing the proto-key $\mathcal{P}_{il}$ for some ancestor v_l of v_j (and descendent of v_i).

In particular, the last property implies that given the knowledge of the key $\mathcal{L}_{ij}$ it is computationally difficult to recover the proto-key $\mathcal{P}_{ij}$.

Once we have defined a way to generate a family of proto-keys featuring the above properties (which we will call a "proto-key assignment"), it is possible to make available to each subscriber the $O(N)$ secret keys corresponding to all the subsets he/she belongs to, by giving him/her only $O(\log^2 N)$ proto-keys, as described below.

Let u be the leaf representing the user within the tree $\mathcal{T}_{SD}$ and let $r_\mathcal{T}$ be the root of $\mathcal{T}$. Furthermore, let $r_\mathcal{T} \equiv u_0, u_1, \ldots, u_t \equiv u$ be all the ancestors of u on the path from $r_\mathcal{T}$ down to u, and denote by s_h the sibling of u_h, $h = 1, \ldots, t$.

By definition, the subtree difference sets S_{ij} containing u are precisely those whose primary root v_i is one of the u_h's and whose secondary root v_j is a descendent[3] of $s_{h'}$ for some $h' > h$.

For instance, among the subsets whose primary root is $r_\mathcal{T}$, the ones containing u are those whose secondary root v_j is a descendent of some s_h. Notice that, by the first property of the proto-keys assignment described above, to compute the key $\mathcal{L}_{r_\mathcal{T} v_j}$ corresponding to such subset, it is enough to know the proto-key $\mathcal{P}_{r_\mathcal{T} v_j}$, which in turn (for the second property) can be obtained from the proto-key $\mathcal{P}_{r_\mathcal{T} s_h}$; thus, by giving the

[3]For our purposes, a node v will be considered among its own descendants.

user the $t = \log N$ proto-keys $\mathcal{P}_{r_\mathcal{T} s_1}, \ldots, \mathcal{P}_{r_\mathcal{T} s_t}$, he/she will be able to efficiently compute the keys relative to all the subsets $S_{r_\mathcal{T} v_j}$ he/she belongs to.

Repeating the same reasoning for all the $\log N$ ancestor u_h of u, we can conclude that $O(\log^2 N)$ proto-keys suffice to allow the user u to recover all the $O(N)$ relevant keys.

To extend the SD scheme to the asymmetric scenario, one would like to generalize the basic idea used for the case of the CS method: namely, define an ID mapping for all the subsets $S_{ij} \in \mathcal{S}_{SD}$ and then employ an IBE scheme to extract all the relevant private keys. However, as already observed, to avoid an explosion of the user's storage, it is necessary to use a scheme satisfying the characteristic properties of a "proto-key assignment," which ordinary IBE schemes do not seem to support, for this would require the capability of deriving "children" proto-keys from a given proto-key. Luckily, the more powerful notion of general Hierarchical Identity-Based Encryption (*cf.* Chapter 2, Section 2.4.2) offers the functionalities needed, leading to the solution described below.

At a high level, our construction proceeds as follows. First, we define a (non-binary) tree $\mathcal{T}'_{SD}$, whose non-leaf vertices[4] are associated in a natural way to nodes in $\mathcal{T}_{SD}$, and whose leaves correspond to subsets of the cover family $\mathcal{S}_{SD}$. Then, we will decorate $\mathcal{T}'_{SD}$ with a labeling, which in turn will induce a mapping HID associating a hierarchical identifier to all vertices in $\mathcal{T}'_{SD}$. Using a HIBE scheme, we can then associate a private key to each vertex by first assigning the master secret key $\mathsf{master}_{\mathsf{HIBE}}$ to the root of $\mathcal{T}'_{SD}$, and then recursively associating to a vertex w the private key resulting by running the $\mathsf{HIBE.Extract}$ algorithm (with w's parent 's private key) on the label of the edge going into w.

The tree $\mathcal{T}'_{SD}$ is constructed from the tree $\mathcal{T}_{SD}$ in the following two steps:

1. For each internal node u in $\mathcal{T}_{SD}$, add a vertex w (associated to u) as a child to the root of $\mathcal{T}'_{SD}$, and mark it as a non-leaf. Therefore, at $Level_1$ of the tree $\mathcal{T}'_{SD}$ there

[4]For the sake of clarity, we will refer to the nodes of $\mathcal{T}'_{SD}$ as *vertices*, so as to distinguish them from the *nodes* of $\mathcal{T}_{SD}$.

are exactly $N-1$ vertices, corresponding to all the possible *primary roots* u_i of a generic subset S_{ij}.

2. For each vertex w at $Level_1$ of $\mathcal{T}'_{SD}$ (and recursively for all non-leaf vertices at lower levels):

   (a) if w is associated to a leaf u in $\mathcal{T}_{SD}$, then add a single child to w in $\mathcal{T}'_{SD}$, and mark it as a leaf;

   (b) otherwise, w is associated to an internal node u in $\mathcal{T}_{SD}$. Let u^ℓ and u^r denote respectively the left and right children of node u in $\mathcal{T}_{SD}$. Then, add to w: 1) a non-leaf child w^ℓ associated to u^ℓ; 2) a non-leaf child w^r associated to u^r; and 3) another child, marked as a leaf.

Now, each leaf vertex w in $\mathcal{T}'_{SD}$ corresponds to a subset of the cover family $\mathcal{S}_{SD}$ as follows. Consider the path p from the root of $\mathcal{T}'_{SD}$ down to such a leaf w. Let w_i be the second vertex in the path p (i.e., w_i is a vertex at $Level_1$), and w_j be the second-to-last vertex in p (i.e. w_j is the parent of w). Then, the subset corresponding to w is the subset S_{ij}, whose primary root is the node u_i in $\mathcal{T}_{SD}$ associated to vertex w_i, and whose secondary root is the node u_j in $\mathcal{T}_{SD}$ associated to vertex w_j.

To label $\mathcal{T}'_{SD}$, we will reuse the ID$(\cdot)$ mapping that we introduced in discussing the public-key extension of the CS method. Recall that ID$(\cdot)$ maps a node u in $\mathcal{T}_{SD}$ into a bitstring of 0's and 1's, according on u's position within $\mathcal{T}_{SD}$. Then, we can label the edges of T'_{SD} as follows:

1. label all edges that lead into leaves of T'_{SD} with $\perp$;

2. label an edge from the root of T'_{SD} to a vertex w at $Level_1$ with the identifier ID(u) corresponding to the node u in T_{SD} associated to vertex w;

3. label all remaining edges with 0 or 1, depending on whether the edge is of the form (w, w^ℓ) or (w, w^r) (*cf.* step 2b. in the definition of $\mathcal{T}'_{SD}$).

Given such labeling of the edges of $\mathcal{T}'_{SD}$, we can define the hierarchical identifier $\mathrm{HID}(w)$ of a vertex w of $\mathcal{T}'_{SD}$ as the *sequence* of label read off the path from the root of $\mathcal{T}_{SD'}$ down to w. In other words:

1. if w is a vertex at $Level_1$, then $\mathrm{HID}(w)$ is the length-1 sequence $\langle \mathrm{ID}(u) \rangle$, where u is the node in $\mathcal{T}_{SD}$ associated to w;

2. if w is an internal vertex at a deeper level, let w' be w's ancestor at $Level_1$, and u and u' the nodes in $\mathcal{T}_{SD}$ associated respectively to w and w'. By construction of $\mathcal{T}'_{SD}$, u is a descendent of u' in T_{SD}, so that $\mathrm{ID}(u')$ is a prefix of $\mathrm{ID}(u)$: $\mathrm{ID}(u) = \mathrm{ID}(u') \circ b_1 \circ \ldots \circ b_h$, for some $h \in [1, \log N]$. Then:

$$\mathrm{HID}(w) = \langle ID(u'), b_1, \ldots, b_t \rangle.$$

3. if w_l is a leaf in $\mathcal{T}'_{SD}$, and w is its parent, and $\mathrm{HID}(w) = \langle ID(u'), b_1, \ldots, b_t \rangle$, then:

$$\mathrm{HID}(w_l) = \langle ID(u'), b_1, \ldots, b_t, \perp \rangle.$$

Notice that by the correspondence of subsets of $\mathcal{S}_{SD}$ with leaves of T'_{SD}, the mapping $\mathrm{HID}(\cdot)$ effectively applies to the S_{ij}'s as well.

Given this definition of the $\mathrm{HID}(\cdot)$ mapping, we can describe the initialization $\mathsf{BE.Setup}$ for the public-key SD method as follows: the Center runs the $\mathsf{HIBE.Setup}$ algorithm of an HIBE scheme and publishes $\mathsf{params}_{\mathsf{HIBE}}$ and a description of the mapping $\mathrm{HID}(\cdot)$ as the public key for the SD-based broadcast encryption scheme. Notice that we have managed to produce a compressed representation of the public keys of all the subsets of $\mathcal{S}_{SD}$ in $O(1)$ space.

Now, the distribution of the secret decryption data to the subscribers can be carried out as another instantiation of the proto-keys assignment: The key $\mathcal{L}_{ij}^{\mathrm{PRI}}$ relative to a given subset S_{ij} will be the private key extracted from the public key $\mathcal{L}_{ij}^{\mathrm{PUB}} = \mathrm{HID}(S_{ij})$.

Formally:[5]

$$\mathcal{L}_{ij}^{\textsc{Pri}} \leftarrow \mathsf{HIBE.Extract}(\mathsf{params}_{\mathsf{HIBE}}, \langle \mathrm{ID}(u_i), b_1, \ldots, b_h, \perp \rangle, \mathcal{P}_{ij})$$
$$\mathcal{P}_{ij} \leftarrow \mathsf{HIBE.Extract}(\mathsf{params}_{\mathsf{HIBE}}, \langle \mathrm{ID}(u_i), b_1, \ldots, b_h \rangle, \mathcal{P}_{il})$$
$$\ldots$$
$$\mathcal{P}_{ii} \leftarrow \mathsf{HIBE.Extract}(\mathsf{params}_{\mathsf{HIBE}}, \langle \mathrm{ID}(u_i) \rangle), \mathsf{master}_{\mathsf{HIBE}})$$

where $\mathrm{ID}(u_j) = \mathrm{ID}(u_i) \circ b_1 \circ \ldots \circ b_h$, u_l is the parent of u_j, and $\mathsf{master}_{\mathsf{HIBE}}$ is the master secret key output by the Setup algorithm and known only to the Root of the HIBE, role that in our setting is played by the Center.

From the above definitions, it is clear that the first two properties of a proto-key assignment are fulfilled; moreover, the validity of the third one follows from the security of the HIBE scheme, that ensures the computational difficulty of obtaining a private key for any identifier without knowing the private key of any ancestor lying higher in the hierarchy of the system.

Direct consequence of the application of the proto-key assignment to the public-key extension, is that each subscriber only needs to store $O(\log^2 N)$ proto-keys, as in the original symmetric-key SD method. As for the communication complexity, the cover finding algorithm characteristic of the SD method ensures that $2r - 1$ ciphertexts will suffice in the worst case to broadcast the session key to all the privileged users in the system.

Security. The formal IND-ID-CCA1-security of the scheme again follows almost immediately from the powerful security definition of HIBE. Indeed, when revoking some set $\mathcal{R}$ of users, none of the proto-keys the adversary learns is an ancestor of any of the hierarchical identifies corresponding to the sets covering $\mathcal{N} \backslash \mathcal{R}$. This property is fairly easy to verify, and a simple hybrid argument will again complete the security proof.

A Concrete Instantiation. When used in conjunction with the HIBE of [16] the

[5]We remark that the values of keys and proto-keys are not uniquely defined by these probabilistic assignments. In particular, deriving the value of the "same" key twice from some of its ancestors will likely result in different keys. However, any value we get is equally functional by the definition of HIBE.

asymmetric variation of the SD scheme proposed above requires users to store $O(\log^2 N)$ keys, and it entails communication complexity $O(r)$, matching the efficiency parameters of the symmetric-key SD method.

4.6 Public-Key Extension of the LSD Method

As described in Chapter 3, Section 3.3, the LSD scheme [51] only differs from the SD method of [65] for the use of a smaller subcollection $\mathcal{S}_{LSD}$ of the Subset-Cover family $\mathcal{S}_{SD}$. We can extend it to the asymmetric setting applying exactly the same idea used to generalize the SD method to the public-key scenario: indeed, any HIBE scheme can be employed to distribute the necessary proto-keys to the users of the system, according to the same label-distribution strategy defined for the original LSD scheme in its conventional symmetric mode.

A Concrete Instantiation. As for the efficiency parameters of such public-key extension, we can repeat the same discussion outlined for the SD scheme: namely, if we use the HIBE proposed in [16] (which is currently the best known implementation), the public-key extension maintains the same communication complexity and requires user to store the same number of keys as in the original, symmetric LSD scheme.

4.6.1 Inclusion-Exclusion Trees

Our extension to the public-key setting can be coupled with the use of Inclusion-Exclusion Tree (IE-Trees) method of [51] (*cf.* also Chapter 3, Section 3.3) in the case of both the SD scheme and the LSD scheme, since once a cover of the set of privileged users has been obtained, both the encryption and the decryption steps can be performed making use of our HIBE-based technique presented above.

Chapter 5

Forward-Secure Public-Key Broadcast Encryption

5.1 Introduction

As observed in Chapter 3 (Section 3.1), one of the most challenging variants of public-key broadcast encryption schemes is the one for *stateless receivers* [67, 65, 80, 34, 36, 58]. In this setting, there is a single "short" public key associated with the system, while each active user has his own distinct secret key (given to this user, at the time he joins the system, by a trusted Center who also generates the public key). Since the receivers are stateless, their secret keys cannot be updated, and the transmission ciphertext should explicitly contain all the information that a legal user needs to decrypt the message with his secret key. Specifically, the encryption algorithm only depends on the message to be encrypted, the public key, and the set $\mathcal{R}$ of "revoked" users: precisely all the users outside $\mathcal{R}$ should have the power to decrypt the message, without any need to listen to prior transmissions, nor to always be "on-line", etc.

The above restrictions of the stateless receiver scenario pose at least the following two limitations which we address in this work:

1. broadcast encryption schemes are specifically designed and optimized for the case

when the user population is large, while the set of revoked users $\mathcal{R}$ is considerably smaller. Indeed, the complexity of such schemes is typically (and necessarily in the stateless receiver setting) proportional to the number of revoked users $r = |\mathcal{R}|$. Unfortunately, this creates huge efficiency problems not only when the system has run for a while and accumulated a lot of "revoked" users, but also at the time of system initialization, when the number of users is still small. A natural "solution" to this problem would be to pretend that the "non-existing" users are actually not yet revoked. However, this means that when such a "non-existent" user actually appears and gets his secret key, he can potentially understand all the previous broadcasts, which he was not supposed to get (*i.e.*, did not pay for). Clearly, it would be desirable to be able to give such users secret keys which only let them understand future, but not *past* transmissions.

2. Assuming a legal user (unintentionally) exposed his key at some time period, the attacker can read all the transmissions which included this user — both in the "future" and in the "past". While in the stateless receiver scenario such exposure inevitably jeopardizes all future transmissions, it would be nice to devise a key-evolution mechanism that at least protects *past* transmissions.

Both limitations above reduce to the same problem of *forward-secrecy* [3, 6, 26]: exposing the secret key of user u for some "time period" t, should leave the transmissions for all prior periods $t' < t$ secure, even if u was *not* revoked in these prior transmissions. In other words, assuming there is a globally-known notion of time, and that the encryption algorithm can take the "current" time t as an extra parameter, each user should seek to somehow evolve his secret key at end of each time period so that: (1) he can still decrypt all ciphertexts for the current time period (as long as he is not revoked); and (2) exposing his current secret key will not compromise any of the ciphertexts for previous time periods (even those where he was not revoked).

Similar to the signature and (regular) encryption scenarios, we call the resulting broadcast encryption schemes *forward-secure* (FSBE). We remark, however, that FSBE schemes seem to be much more useful and motivated than regular forward-secure encryption schemes (FSE). Indeed, attacking the security of the communication of a stad-alone user requires a very focused and self-motivated adversary, for a handful of reasons. First, the user may properly guard his secret key, and the life of the adversary could be much easier if she could monitor several users, and then try to compromise the least careful of them. Second, ciphertexts meant for a single user are usually not widely available, so that the adversary would have to take great care to intercept and record past ciphertexts. Moreover, before compromising the secret key, the adversary cannot tell which ciphertexts are "important" to her goal, so that an adversary trying to steal a business secret from a user could have to store along all the encrypted gossips the user exchanged with his friends. Finally, there are arguably not that many scenarios where past messages to some stand-alone user were anyway really "profitable" to an attacker. In contrast, broadcast encryption schemes are typically used in commercial settings, where: (1) the content provider could suffer a significant loss if many users do not pay for the service; and (2) correspondingly, there could actually be profit associated with illegal deciphering of past transmissions, as they could later be sold as bootleg copies. Additionally, the broadcast scenario by definition gives anybody easy access to the encrypted content, and thus there is no problem obtaining "important past ciphertexts". Finally, since the content is encrypted for many users, the attacker has many avenues for compromising secret keys. To summarize, the problem of designing efficient FSBE schemes is well motivated and still stays within the realm of the stateless receiver scenario, since users can evolve their secret keys independently from each other and from prior transmissions.

5.2 Our Results

In this chapter, based on work first published as [82], we construct the first FSBE scheme. It resists the strongest type of *chosen ciphertext* and *key corruption* attack, where the adversary is allowed to corrupt users (thus obtaining their secret keys) in any order, and can also ask non-corrupted users to decrypt any ciphertexts of her choice, during any time period. The security of our scheme is based on the decisional bilinear Diffie-Hellman inversion (DBDHI) assumption, in the standard model.

We show how any PHIBE can be used to build a secure FSBE. For that, we define another general notion of independent interest: the *Access Hypergraph framework*. In this framework, one considers hypergraphs where vertices corresponds to users having secret keys, and hyperedges (*i.e.*, subsets of vertices) corresponds to possible public keys. These keys are then used to encrypt messages via a *Hypergraph Encryption* scheme. For example, in the broadcast encryption (BE) setting (*cf.* Chapter 4, Section 4.3) the hyperedges consist of all sets of $(N - r_{max})$ users, where N is the total number of users and r_{max} is the maximum number of revoked users. Encrypting to a set S corresponds to the fact that we want precisely the users in S to understand the transmissions. Similarly, forward-secure encryption (FSE) scheme (*cf.* Chapter 2, Section 2.5) corresponds to sets $\{1\}, \{1, 2\}, \ldots, \{1, \ldots, T\}$, where T is the total number of time periods. Indeed, encrypting the message at time period 2 (*i.e.*, to a set $\{1, 2\}$) means that only "current and past periods" 1 and 2 can understand the transmission, but "future periods" $3, \ldots, T$ cannot.

We then formalize a notion of *reduction* between various access hypergraphs and observe that the FSE construction of [26] and the BE construction of [34] (though quite different), can both be formally viewed as reductions between the corresponding FSE/BE-access hypergraph and HIBE-access hypergraph. Additionally, we observe that the access hypergraph for FSBE corresponds to a *hypergraph product* of FSE and BE access hypergraphs, while PHIBE corresponds to the hypergraph product of two independent HIBE-access hypergraphs. Since we show that our reduction operation is closed under hypergraph product, we get a non-trivial reduction between FSBE and PHIBE (in particular, this reduction would be very hard to observe directly, but it easily follows from our modular approach). Thus, the PHIBE we construct immediately yields the desired FSBE.

To summarize, in addition to the first construction of FSBE, we also define and utilize the notion of access hypergraph and reductions between access hypergraphs, which allows us to present our results in a modular way, and unifies several previous results in the literature.

Parameters. Our construction obtains a secure FSBE scheme from any secure PHIBE at the cost of increasing: the *storage complexity* (*i.e.*, the size of each user's secret information) by a factor of $O(\log T \log^2 N)$; the *communication complexity* (*i.e.*, the size of the encrypted broadcast) by a factor of $2r$; the *encryption complexity* (*i.e.*, the running time of the encryption algorithm) by a factor of $2r$ plus an additional $O(\log T + r \log N)$ overhead, (where T is the maximum number of instants, N is size of the users universe and r is the *actual* number of revoked users.)

Alternatively, using the specific construction for fs-HIBE of [16] we achieve *storage complexity* $O(\log^2 N(\log T + \log N))$, *encryption complexity* $O(r(\log T + \log N))$ and *communication complexity* $O(r \log T)$.

Why this is non-trivial. Not surprisingly, our FSBE scheme utilizes a combination of ideas from the design of regular broadcast encryption (BE) and forward-secure encryption schemes. At first glance, one might be tempted to think that building FSBE from FSE and BE, or PHIBE — from two HIBE's, or for that matter, any encryption for a product of two access hypergraphs — from that of the corresponding base hypergraphs, might be a simple task which can be accomplished generically. Unfortunately, this does not seem to be the case. Let us illustrate this for the case of FSBE, the others are similar.

Specifically, one might be tempted to construct a FSBE scheme from any BE and FSE schemes by some kind of simple black-box composition, where user u at time t would have his "BE-key" k_b and "FSE-key" k_f and the message will be encrypted with respect to both of these keys. Unfortunately, this solution can never be secure. Indeed, corrupting user u at any time $t > 1$ and user v at time 1, we will obtain user u's entire secret key at time period 1. And this is wrong, since, for example, it allows to decrypt a message encrypted at time 1 to a set of users containing u but not v.

Chosen-Ciphertext Security. For simplicity, the presentation below will focus on the simpler notion of chosen-plaintext (IND-ID-CPA) security, as the IND-ID-CPA setting already contains most of the technical challenges we have to deal with. However, all our definitions and constructions naturally extend to chosen-ciphertext (IND-ID-CCA) security. This is elaborated in Section 5.6.

Organization. In Section 5.3, we present a formal model for Forward-Secure Broadcast Encryption (FSBE), specifying syntax and FSBE.IND-ID-CPA security. In Section 5.4, we introduce the Access Hypergraph framework, which in particular includes the definition of Hypergraph Encryption (HE) scheme and the corresponding notion of HE.IND-ID-CPA security. We then develop some technical lemmas, leading to an approach to construct Forward-Secure Broadcast Encryption schemes out of the *product* of Access Hypergraphs. We conclude by showing how PHIBE provides the last ingredient to obtain the first efficient Forward-Secure Broadcast Encryption scheme.

5.3 Formal Model

In this section, we present a model for public-key forward-secure broadcast encryption. Following standard practice [65, 80, 36], we assume that the actual content of the broadcast is encrypted with a random session key, using a semantically-secure symmetric-key cipher; thus, the goal of a broadcast encryption scheme is to *encapsulate* the session key within an *enabling block* in such a way that only the authorized users will be able to retrieve the session key, and thus recover the broadcast content.

5.3.1 FSBE: Syntax

Definition 31 (FSBE: Forward-Secure Broadcast Encryption Scheme). A FSBE scheme $\mathcal{E}_{\mathsf{FSBE}}$ is a five-tuple of probabilistic polynomial-time algorithms (Setup, $\mathsf{Register}$, Update, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$), where:

- Setup, the *key generation algorithm*, is used by the Center to set up the parameters of the scheme. Setup takes as input a security parameter 1^λ and possibly $1^{r_{\max}}$ (where $r_{\max}$ is a revocation threshold, *i.e.* the maximum number of users that can be revoked). The input also includes the total number N of users in the system and the total number of time periods T. Setup generates the public key $\mathsf{params}_{\mathsf{FSBE}}$ and the master secret key $\mathsf{master}_{\mathsf{FSBE}}$. The Center publishes $\mathsf{params}_{\mathsf{FSBE}}$ and keeps $\mathsf{master}_{\mathsf{FSBE}}$ secret.

- $\mathsf{Register}$, the *registration algorithm*, is used by the Center to compute the secret initialization data for a new user. $\mathsf{Register}$ takes as input the master secret key $\mathsf{master}_{\mathsf{FSBE}}$, the identity u of the user and the current time period $t < T$, and outputs the new secret key $\mathsf{SK}_{u,t}$.

- Update, the *key update algorithm*, takes as input the public key $\mathsf{params}_{\mathsf{FSBE}}$, the identity u of a user, the current time period $t < T$ and the user's secret key $\mathsf{SK}_{u,t}$. It outputs the new secret key $\mathsf{SK}_{u,t+1}$ valid for the subsequent time period $t + 1$.

- $\mathsf{Encrypt}$, the *encryption algorithm*, is used to encapsulate a given session key s within an enabling block $\mathcal{B}$. $\mathsf{Encrypt}$ takes as input the public key $\mathsf{params}_{\mathsf{FSBE}}$, the session key s, the current time period t and a set $\mathcal{R}$ of revoked users (with $|\mathcal{R}| \leq r_{\max}$, if a threshold has been specified to the Setup algorithm) and returns the enabling block $\mathcal{B}$ to be broadcast.

- $\mathsf{Decrypt}$, the *decryption algorithm*, is a deterministic algorithm that takes as input the public key $\mathsf{params}_{\mathsf{FSBE}}$, the identity u of a user, a time period $t < T$, the user's secret key $\mathsf{SK}_{u,t}$ and an enabling block $\mathcal{B}$. $\mathsf{Decrypt}$ returns a session key s or the special rejection symbol $\perp$.

A FSBE scheme $\mathcal{E}_{\mathsf{FSBE}}$ should satisfy the following correctness constraint: for any pair $(\mathsf{params}_{\mathsf{FSBE}}, \mathsf{params}_{\mathsf{FSBE}})$ output by $\mathsf{FSBE.Setup}(1^\lambda, 1^{r_{\max}}, N, T)$, any $t < T$, any $\mathcal{R} \subseteq \mathcal{N}, (|\mathcal{R}| \leq r_{\max})$, any user $u \in \mathcal{N} \setminus \mathcal{R}$ with secret key $SK_{u,t}$ (properly generated for time period t) and any session key s:

$$s = \mathsf{FSBE.Decrypt}(\mathsf{params}_{\mathsf{FSBE}}, u, t, \mathsf{SK}_{u,t}, \mathsf{FSBE.Encrypt}(\mathsf{params}_{\mathsf{FSBE}}, \mathcal{R}, t, s)).$$

5.3.2 FSBE: Security

Once a user leaks its secret key, the security of every subsequent broadcast communication is compromised, as long as such user is not revoked. Intuitively, forward-security for broadcast encryption schemes guarantees that this is the only case in which unauthorized access to the broadcast content can occur; in other words, decryption of a ciphertext produced at time t succeed with non-negligible probability only given the secret key generated for an authorized user in an instant prior to t. Even corruption of many users (possibly at different time instants), does not help in recovering the content broadcast in some instant t, if all the users corrupted before t are considered revoked for that broadcast.

A FSBE scheme $\mathcal{E}_{\mathsf{FSBE}}$ is forward-secure against chosen-ciphertext attack if for all polynomial N and T, no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage in the following game:

Setup: The challenger runs algorithm $\mathsf{FSBE.Setup}(1^\lambda, 1^{r_{\max}}, N, T)$; it then gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{FSBE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{FSBE}}$ secret.

Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. Corrupt(u, t): the challenger runs algorithm $\mathsf{FSBE.Register}(\mathsf{master}_{\mathsf{FSBE}}, u, t)$ to generate the private key $\mathsf{SK}_{u,t}$ corresponding to user u at time instant t, and sends $\mathsf{SK}_{u,t}$ to $\mathcal{A}$.

2. Decrypt$(u, t, \mathcal{B}_j)$: the challenger runs algorithm $\mathsf{FSBE.Register}(\mathsf{master}_{\mathsf{FSBE}}, u, t)$ to recover the private key $\mathsf{SK}_{u,t}$ corresponding to user u at time instant t. It then runs algorithm $\mathsf{FSBE.Decrypt}(\mathsf{params}_{\mathsf{FSBE}}, u, t, \mathsf{SK}_{u,t}, \mathcal{B}_j)$ and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs a time instant t^* along with two equal length session keys $s_0, s_1 \in \mathcal{M}$ on which it wishes to be challenged. The challenger picks a random bit $b \in \{0, 1\}$, sets $\mathcal{B}^* \doteq \mathsf{FSBE.Encrypt}(\mathsf{params}_{\mathsf{FSBE}}, \mathcal{R}_{t^*}, t^*, s_b)$, where $\mathcal{R}_{t^*} \doteq \{u \mid \mathcal{A}$ asked a query $\textsc{Corrupt}(u, j)$, for some $t \leq t^*\}$. The challenger sends $\mathcal{B}^*$ as a challenge to $\mathcal{A}$.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is one of the following:

1. $\textsc{Corrupt}(u, t)$: the challenger first checks that $t > t^*$, and if so, it responds as in **Phase 1**. Notice that if a bound r_{max} was specified in Setup, then the adversary is restricted to corrupt at most r_{max} distinct users via $\textsc{Corrupt}(\cdot, \cdot)$ queries.

2. $\textsc{Decrypt}(u, t, \mathcal{B}_j)$: the challenger first checks that either $\mathcal{B}_j \neq \mathcal{B}^*$ or $u \in \mathcal{R}_{t^*}$ or $t \neq t^*$ and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0, 1\}$ and wins the game if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as an FSBE.IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the FSBE scheme $\mathcal{E}_{\mathsf{FSBE}}$ as:

$$\mathsf{Adv}_{\mathsf{FSBE},\mathcal{A}} = \left| \Pr[b = b^*] - \frac{1}{2} \right| .$$

The probability is over the random bits of the challenger and of the adversary.

We can also define the notion of selective-identity chosen-ciphertext security for FSBE (FSBE.IND-sID-CCA). The game is exactly as for the FSBE.IND-ID-CCA case, except that the adversary $\mathcal{A}$ discloses to the challenger the target instant t^* and the set $\mathcal{R}^*$ of revoked users *before* the **Setup** phase. Thus, the restriction on CORRUPT queries from **Phase 2** also holds in **Phase 1**.

Definition 32 (CHOSEN-CIPHERTEXT SECURITY FOR FSBE)**.**
An FSBE scheme $\mathcal{E}_{\mathsf{FSBE}}$ is $(\tau, q_{\text{ID}}, \varepsilon_{\mathsf{FSBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure if for any τ-time FSBE.IND-ID-CCA (resp. FSBE.IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen CORRUPT queries and at most q_C chosen DECRYPT queries, it holds that $\mathsf{Adv}_{\mathsf{FSBE},\mathcal{A}} < \varepsilon_{\mathsf{FSBE}}$.

We define chosen-plaintext security for an FSBE scheme according to an attack game that is exactly as in the preceding game, except that the adversary is not allowed to issue DECRYPT queries. The adversary may still issue adaptive chosen CORRUPT queries.

Definition 33 (Chosen-Plaintext Security for FSBE).
An FSBE scheme $\mathcal{E}_{\mathsf{FSBE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{FSBE}})$-IND-ID-CPA (resp. IND-sID-CCA)-secure if it is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{FSBE}})$-IND-ID-CCA (resp. IND-sID-CCA)-secure.

5.4 The Access Hypergraph Framework

In some applications of public-key encryption, the intended recipient of the encrypted content is not a *single* entity, but rather a *group* of entities. In such scenario, each entity is associated with its own specific secret key, whereas public keys (which are used to encrypt content) corresponds to set of these entities. Typical examples of this kind of scenarios are Broadcast Encryption (in which entities are users), Forward Encryption (entities being time instants), Hierarchical Identity-Based Encryption and possibly others.

In this section, we introduce a unifying framework based on the notion of *hypergraph*, which enables a uniform treatment of scenarios of the type described above. To emphasize our use of hypergraphs as structures to regulate access to encrypted content, we will refer to such scenarios using the terminology of *access hypergraphs*. For more information about hypergraphs, see [8].

After defining the notion of *hypergraph encryption* scheme for an access hypergraph, we will show how any two such structures can be coupled to yield a combined access hypergraph inheriting all their properties. As a corollary, this will provide a way to obtain an efficient construction of Forward-Secure Broadcast Encryption as a combination of Broadcast Encryption and Forward-Secure Encryption.

Definition 34 (Hypergraph).
A hypergraph $\mathcal{H}$ is a pair (V, S) where V is a finite set and S is a family of subsets of V *i.e.*, $S \subseteq \mathcal{P}(V)$. The elements of V are called *vertices* and the elements of S are called *hyperedges* (or *sets*). The hypergraph $\mathcal{H}$ implicitly defines an *incidence* function $f : V \to \mathcal{P}(S)$, which associates each vertex $v \in V$ to the family of sets in S containing

v; formally,

$$f(v) \doteq \{s \in S \mid v \in s\}.$$

As argued above, our motivation for introducing access hypergraph is to support an encryption mechanism such that a single ciphertext could be decrypted by several entities. We formalize this with the notion of *hypergraph encryption* scheme.

Definition 35 (Hypergraph Encryption Scheme)**.**
A HE scheme $\mathcal{E}_{\mathsf{HE}}$ for an access hypergraph $\mathcal{H} = (V, S)$ is a triple of polynomial-time algorithms (Setup, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$), where:

- Setup, the *key generation* algorithm, takes as input a security parameter 1^λ and outputs some global parameters $\mathsf{params}_{\mathsf{HE}}$ (which describes some simple mapping associating identifiers to hyperedges of $\mathcal{H}$), and a sequence of secret keys $\mathsf{master}_{\mathsf{HE}} = \langle \mathsf{SK}_v \rangle_{v \in V}$ which associates a secret key to each vertex of $\mathcal{H}$.[1]

- $\mathsf{Encrypt}$, the *encryption algorithm*, is a probabilistic algorithm that takes as input the public parameters $\mathsf{params}_{\mathsf{HE}}$, a set $s \in S$ and a message m and outputs the corresponding ciphertext C.

- $\mathsf{Decrypt}$, the *decryption algorithm*, is a deterministic algorithm that takes as input the secret key SK_v of a vertex $v \in V$, a set $s \in S$ and a ciphertext C and returns either a message m or the special rejection symbol $\perp$.

A HE scheme $\mathcal{E}_{\mathsf{HE}}$ should satisfy the following correctness constraint: for any pair $(\mathsf{params}_{\mathsf{HE}}, \mathsf{master}_{\mathsf{HE}})$ output by $\mathsf{HE.Setup}$, any set $s \in S$, any vertex $v \in s$ and any message m:

$$m = \mathsf{HE.Decrypt}(\mathsf{params}_{\mathsf{HE}}, s, \mathsf{SK}_v, \mathsf{HE.Encrypt}(\mathsf{params}_{\mathsf{HE}}, s, m)).$$

[1]Specific instances of hypergraph encryption schemes may provide some compact description of $\mathsf{master}_{\mathsf{HE}}$ (*e.g.*, via a single master secret key), as long as such "compression" does not jeopardize the security requirements stated in Definition 37 below.

A HE scheme $\mathcal{E}_{\mathsf{HE}}$ is secure against chosen-ciphertext attack if no polynomial-time adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:
Setup: The challenger runs algorithm $\mathsf{HE.Setup}(1^\lambda)$; it then gives $\mathcal{A}$ the resulting system public key $\mathsf{params}_{\mathsf{HE}}$ and keeps the master secret key $\mathsf{master}_{\mathsf{HE}}$ secret to itself.
Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where each q_j is one of the following:

1. $\textsc{Corrupt}(v)$: the challenger hands off to $\mathcal{A}$ the secret key SK_v corresponding to vertex $v \in V$.

2. $\textsc{Decrypt}(v, s, C_j)$: the challenger first picks keys SK_v from $\mathsf{master}_{\mathsf{HE}}$ and then runs $\mathsf{HE.Decrypt}(\mathsf{params}_{\mathsf{HE}}, s, \mathsf{SK}_v, C_j)$ and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs a set s^* and two equal length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue the query $\textsc{Corrupt}(v)$ for $v \in s^*$. The challenger picks a random bit $b \in \{0,1\}$, sets $C^* \doteq \mathsf{HE.Encrypt}(\mathsf{params}_{\mathsf{HE}}, s^*, m_b)$, and sends C^* as a challenge to $\mathcal{A}$.
Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$, where each q_j is one of the following:

1. $\textsc{Corrupt}(v)$: the challenger first checks that $v \notin s^*$ and if so, it responds as in **Phase 1**.

2. $\textsc{Decrypt}(v, s, C_j)$: the challenger first checks that either $C_j \neq C^*$ or $s \neq s^*$ and if so, it responds as in **Phase 1**.

Guess: The adversary outputs a guess $b^* \in \{0,1\}$ and wins the game if $b = b^*$.
We refer to such an adversary $\mathcal{A}$ as an $\mathsf{HE.IND\text{-}ID\text{-}CCA}$ adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}_{\mathsf{HE}}$ as:

$$\mathsf{Adv}_{\mathsf{HE},\mathcal{A}} = \left| \Pr[b = b^*] - \frac{1}{2} \right| .$$

The probability is over the random bits of the challenger and of the adversary.

Definition 36 (Chosen-Ciphertext Security for HE).
An HE scheme $\mathcal{E}_{\mathsf{HE}}$ is $(\tau, q_{\mathrm{ID}}, q_C, \varepsilon_{\mathsf{HE}})$-IND-ID-CCA-secure if, for any τ-time HE.IND-ID-CCA adversary $\mathcal{A}$ that makes at most q_{ID} chosen Corrupt queries and at most q_C chosen Decrypt queries, it holds that $\mathsf{Adv}_{\mathsf{HE},\mathcal{A}} < \varepsilon_{\mathsf{HE}}$.

We define chosen-plaintext security for a HE scheme according to an attack game that is exactly as the preceding game, except that the adversary is not allowed to issue Decrypt queries. The adversary may still issue adaptive chosen Corrupt queries.

Definition 37 (Chosen-Plaintext Security for HE).
An HE scheme $\mathcal{E}_{\mathsf{HE}}$ is $(\tau, q_{\mathrm{ID}}, \varepsilon_{\mathsf{HE}})$-IND-ID-CPA secure if $\mathcal{E}_{\mathsf{HE}}$ is $(\tau, q_{\mathrm{ID}}, 0, \varepsilon_{\mathsf{HE}})$-IND-ID-CCA secure.

Efficiency of a Hypergraph Encryption Scheme. In comparing the efficiency of two or more hypergraph encryptions schemes, the three major complexity measures to be considered are: (1) *storage complexity*, which denotes the space required (in the worst case) to store the secret key SK_v of a vertex v; (2) *encryption complexity i.e.*, the (worse-case) running time of the encryption algorithm $\mathsf{HE.Encrypt}$; and (3) *communication complexity*, which refers to the size of ciphertexts produced by the encryption algorithm.

5.4.1 Examples of Access Hypergraphs

The above definitions of access hypergraph and of hypergraph encryption may appear too abstract at first; to make the underlying idea clearer, we now present some examples.
Forward-Secure Encryption. In the Forward-Security model of [6, 26], time is represented as a sequence of instants $\langle 1, \ldots, T\rangle$ (where T is the total number of time periods.) Although our framework does not directly support sequences, we can easily represent the intrinsic order among time instants as follows:

Definition 38 (Forward-Secure Access Hypergraph).
The access hypergraph $\mathcal{H}_{FS} = (V_{FS}, S_{FS})$ for the Forward-Security model is defined as $V_{FS} \doteq \{1, \ldots, T\}$ and $S_{FS} \doteq \{\{1\}, \{1, 2\}, \ldots, \{1, 2, \ldots, T\}\}$.

Broadcast Encryption. The Broadcast Encryption setting [43, 65] can be described by specifying a universe of users $\mathcal{N}$ and (possibly) a revocation threshold $r_{max} \leq N$, where $N = |\mathcal{N}|$. In our framework, this can be formalized with the following definition:

Definition 39 (Broadcast Access Hypergraph)**.**
The access hypergraph $\mathcal{H}_{BE} = (V_{BE}, S_{BE})$ for the Broadcast Encryption model is defined as $V_{BE} \doteq \mathcal{N}$ and $S_{BE} \doteq \{s \subseteq \mathcal{N} : |s| \geq N - r_{max}\}$.

HIBE **as Hypergraph Encryption for a Family of Access Hypergraphs.** As last example, we show how *Hierarchical ID-Based Encryption* (HIBE) can be seen as a hypergraph encryption for Tree Access Hypergraphs, a special kind of access hypergraph which closely resembles the hierarchical structure of HIBE.

Definition 40 (Tree Access Hypergraph)**.**
Let $\mathcal{T}$ be a tree on a set of nodes N and let $\preceq$ be the "ancestor-descendant" relationship of $\mathcal{T}$. The access hypergraph $\mathcal{H}_{\mathcal{T}} = (V_{\mathcal{T}}, S_{\mathcal{T}})$ associated to $\mathcal{T}$ is defined as $V_{\mathcal{T}} \doteq N$ and $S_{\mathcal{T}} \doteq \{\{u \in V_{\mathcal{T}} \mid u \preceq v\} : v \in V_{\mathcal{T}}\}$.

Lemma 41. *Let $\mathcal{T}$ be a tree and let $\mathcal{H}_{\mathcal{T}} = (V_{\mathcal{T}}, S_{\mathcal{T}})$ be the corresponding Tree Access Hypergraph. An HE.IND-ID-CPA secure hypergraph encryption scheme for $\mathcal{H}_{\mathcal{T}}$ can be obtained from any HIBE.IND-ID-CPA secure HIBE scheme (with essentially the same efficiency parameters of the corresponding HIBE scheme).*

Proof. The $\mathsf{HE.KeyGen}$ algorithm generates $\mathsf{params_{HE}}$ and $\mathsf{master_{HE}}$ setting $\mathsf{params_{HE}} \doteq \mathsf{params_{HIBE}}$ and $\mathsf{SK}_v \doteq \mathsf{HIBE.Extract}^*(\mathsf{params_{HIBE}}, \mathrm{HID}_v, \mathsf{master_{HIBE}})$, *i.e.*, each node v receives the secret key corresponding to its hierarchical identifier HID_v, and the star in $\mathsf{HIBE.Extract}^*$ denotes the fact that the $\mathsf{HIBE.Extract}$ algorithm may have to be run multiple times, depending on the level at which the hierarchical identifier HID_v appears in $\mathcal{T}$. Notice that this construction allows for a compact representation of $\mathsf{master_{HE}}$ which consists of $\mathsf{master_{HIBE}}$.

As for the encryption and decryption algorithms, $\mathsf{HE.Encrypt}$ and $\mathsf{HE.Decrypt}$, they can be obtained from the corresponding components of HIBE:

$$\mathsf{HE.Encrypt}(\mathsf{params}_{\mathsf{HE}}, s, m) \doteq \mathsf{HIBE.Encrypt}(\mathsf{params}_{\mathsf{HIBE}}, \mathrm{HID}_{\hat{v}}, m)$$
$$\mathsf{HE.Decrypt}(\mathsf{SK}_v, s, C) \doteq \mathsf{HIBE.Decrypt}(\mathsf{params}_{\mathsf{HIBE}}, \mathsf{SK}_{\mathrm{HID}_{\hat{v}}}, C)$$

where: $\hat{v}$ is the minimal element in the set s (w.r.t. the $\preceq$ relationship); $\mathrm{HID}_{\hat{v}}$ denotes the hierarchical identifier associated to $\hat{v}$; and $\mathsf{SK}_{\mathrm{HID}_{\hat{v}}}$ is the secret key corresponding to the hierarchical identifier $\mathrm{HID}_{\hat{v}}$, which node v either has (if $v = \hat{v}$), or can derive from its secret information using the $\mathsf{HIBE.Extract}$ algorithm.

It easy to verify that applying the above transformation to a secure HIBE scheme yields a hypergraph encryption for the Tree Access Hypergraph $\mathcal{H}_{\mathcal{T}} = (V_{\mathcal{T}}, S_{\mathcal{T}})$ corresponding to the given tree-hierarchy $\mathcal{T}$. $\square$

Remark 42. Using the specific construction for HIBE of [48], we get an hypergraph encryption for Tree Access Hypergraphs with *storage*, *encryption* and *communication complexity*, all proportional to the depth d of the tree $\mathcal{T}$.

Remark 43. Roughly speaking, a hypergraph encryption scheme for a Tree Access Hypergraph deals with a “static” hierarchy (the one represented by the tree $\mathcal{T}$), whereas a HIBE scheme works also for hierarchies which are not fixed a priori, but change dynamically as the need arises. However, for most applications of HIBE (such as Forward-Secure Encryption [26] and Public-Key Broadcast Encryption [34]), the additional “dynamic” power provided by HIBE is not needed, and the kind of static hierarchy supported by Tree Access Hypergraphs suffices. For these reasons, the use of a full-blown HIBE scheme in Lemma 41 above is not strictly necessary, and we could as well have based our construction on *Tree Encryption*, a straightforward generalization of the notion of *Binary Tree Encryption* (BTE, [26]) to the case of trees with polynomial (in the security parameter λ) breadth and depth.

5.4.2 Reduction between Access Hypergraphs

One advantage of the access hypergraph framework is that it makes it easier to draw connections between the different settings that can be modeled within it, in particular from a security viewpoint. For example, by modeling the notion of HIBE within the access hypergraph framework, it is possible to spot a structural similarity between the approach of [26], which bases Forward-Secure public-key encryption on *Binary Tree Encryption* (BTE, a variant of HIBE), and the construction of [34], which derives a public-key broadcast encryption scheme from any HIBE: both can be viewed as distinct instances of the same reduction argument.

Definition 44 (Reduction).
A reduction from an access hypergraph $\mathcal{H} = (V, S)$ to an access hypergraph $\mathcal{H}' = (V', S')$, denoted by $\mathcal{H}' \leq \mathcal{H}$, consists of two polynomial-time computable functions $g : S' \to \mathcal{P}(S)$ and $k : V' \to \mathcal{P}(V)$ such that:

$$(\forall v' \in V')(\forall s' \in S')[v' \in s' \Leftrightarrow ((\bigcup_{s \in g(s')} s) \cap k(v')) \neq \emptyset].$$

Additionally, define the *cover size* of the reduction as $G = \max\{|g(s')| : s' \in S'\}$, and the *key size* as $K = \max\{|k(v')| : v' \in V'\}$.

The following theorem highlights the relation between the notion of reduction and the notion of hypergraph encryption, which makes it clear how, in designing a reduction between two access hypergraphs, it is important to minimize the cover size G and the key size K (which, ideally, should both be equal to 1), and to keep the running time of the function g as low as possible.

Theorem 45 (Reduction Theorem). *Let $\mathcal{H}$ and $\mathcal{H}'$ be two access hypergraphs. If $\mathcal{H}$ admits a* HE*.IND-ID-CPA secure hypergraph encryption and (g, k) is a reduction from $\mathcal{H}$ to $\mathcal{H}'$, then $\mathcal{H}'$ also admits a* HE*'.IND-ID-CPA secure hypergraph encryption. Moreover, the efficiency of the hypergraph encryption for $\mathcal{H}'$ is related to that of the hypergraph encryption for $\mathcal{H}$ as follows: the storage complexity increases by at most a factor of K,* *the communication complexity increases by at most a factor of G, and the encryption complexity increases by at most a factor of G, plus the additional overhead to evaluate g.*

Proof. To prove the theorem, it suffices to describe how algorithms $\mathsf{HE}'.\mathsf{Extract}$, $\mathsf{HE}'.\mathsf{Encrypt}$ and $\mathsf{HE}'.\mathsf{Decrypt}$ can be implemented assuming the existence of a hypergraph encryption scheme HE for $\mathcal{H}$ and a reduction (g, k) from $\mathcal{H}$ to $\mathcal{H}'$.

The idea behind the construction of the $\mathsf{HE}'.\mathsf{Extract}$ algorithm is (1) to associate every set $s' \in S'$ with the identifiers of all the sets $s \in g(s')$, and (2) to associate every vertex $v' \in V'$ with the secret keys of all the vertices $v \in k(v')$. Formally, on input the security parameter 1^λ, the $\mathsf{HE}'.\mathsf{Extract}$ algorithm first runs $\mathsf{HE}.\mathsf{Extract}$ to obtain $\mathsf{params}_{\mathsf{HE}}$ and $\mathsf{master}_{\mathsf{HE}}$, and then outputs $\mathsf{params}_{\mathsf{HE}'} \doteq (\mathsf{params}_{\mathsf{HE}}, g)$ and $\mathsf{HE}'.\mathsf{SK}_{v'} \doteq \langle (v, \mathsf{HE}.\mathsf{SK}_v) \rangle_{v \in k(v')}$.

To encrypt a message m for a set s', we simply encrypt it with all the public keys associated to s'. Formally, $\mathsf{HE}'.\mathsf{Encrypt}$ can be defined to output a sequence of pairs whose first component is a set, and whose second component is the corresponding encryption: $\mathsf{HE}'.\mathsf{Encrypt}(\mathsf{params}_{\mathsf{HE}'}, s', m) \doteq \langle (s, \mathsf{HE}.\mathsf{Encrypt}(\mathsf{params}_{\mathsf{HE}}, s, m)) \rangle_{s \in g(s')}$.

To decrypt a ciphertext C meant for a set s', the $\mathsf{HE}'.\mathsf{Decrypt}$ algorithm uses a secret key $\mathsf{HE}'.\mathsf{SK}_{v'}$ as follows: for each entry (s_i, C_i) in C and for each entry $(v_j, \mathsf{HE}.\mathsf{SK}_{v_j})$ in $\mathsf{HE}'.\mathsf{SK}_{v'}$ such that $v_j \in s_i$, decrypt C_i with $\mathsf{HE}.\mathsf{Decrypt}(\mathsf{HE}.\mathsf{SK}_{v_j}, s_i, C_i)$. If all decryptions are consistent, then output the resulting plaintext; otherwise output $\perp$.

Notice that if $v' \in s'$, then by definition of reduction there will be at least one entry in the ciphertext C and one entry in $\mathsf{HE}'.\mathsf{SK}_{v'}$ such that $\mathsf{HE}.\mathsf{Decrypt}(\mathsf{HE}.\mathsf{SK}_{v_j}, s_i, C_i)$ can be run, and if C was well-formed, then all its ciphertext components corresponds to the same plaintext.

A simple reduction argument suffices to show that, if $\mathcal{H}' \leq \mathcal{H}$, then applying the above transformation to a hypergraph encryption scheme for $\mathcal{H}$ yields a hypergraph encryption scheme for $\mathcal{H}'$. $\square$

5.4.3 Two Important Reductions

Forward-Secure Encryption from Tree Access Hypergraphs

We now show how the construction of a Forward-Secure public-key encryption scheme by Canetti *et al.* [26] can be recast as a reduction from a suitable Tree Access Hypergraph.

Lemma 46. *There exists a tree hierarchy $\mathcal{T}_{CHK}$ such that $\mathcal{H}_{FS} \leq \mathcal{H}_{\mathcal{T}_{CHK}}$, with $G = 1$, $K = \log T$ and running time $Time(g_{CHK}) = O(\log T)$, where T is number of time periods specified by $\mathcal{H}_{FS}$.*

Proof. At a high level, Canetti et al.'s construction of a Forward-Secure scheme with $T = 2^{d+1} - 1$ time periods [26] is based on a complete binary tree $\mathcal{T}_{CHK}$ of depth d. Instants from 1 to T are associated with nodes of $\mathcal{T}_{CHK}$ according to a pre-order traversal. More precisely, the root of $\mathcal{T}_{CHK}$ represents instant 1, and inductively, if an internal node v represents instant t, then v's left child represents instant $t+1$ and v's right child represents instant $t+n+1$, where n is the number of nodes in v's left subtree.

As described in Definition 40, such hierarchy $\mathcal{T}_{CHK}$ induces an access hypergraph $\mathcal{H}_{\mathcal{T}_{CHK}} = (V_{CHK}, S_{CHK})$ where V_{CHK} is the set of all the time instants, and S_{CHK} is the set of all the paths from the root to every node in $\mathcal{T}_{CHK}$. We now define a reduction from $\mathcal{H}_{\mathcal{T}_{CHK}}$ to $\mathcal{H}_{FS}$ describing the two functions $g_{CHK} : S_{FS} \to \mathcal{P}(S_{CHK})$ and $k_{CHK} : V_{FS} \to \mathcal{P}(V_{CHK})$. On input $s' = \{1, \dots, t\} \in S_{FS}$, g_{CHK} first determines the node $v_t \in V_{CHK}$ that corresponds to instant t, and considers the set s_t of all vertices on the path from the root of $\mathcal{T}_{CHK}$ to v_t. Then, it outputs the set $\{s_t\}$.

On input $t \in V_{FS}$, k_{CHK} first determines the node $v_t \in V_{CHK}$ that corresponds to instant t and considers the path from the root of $\mathcal{T}_{CHK}$ to v_t. Then, it outputs the set containing v_t and all the vertices hanging-off to the right of such path (in other words, k_{CHK} outputs the set containing v_t along with the right sibling of every node of such path.)

It can be easily verified that the above construction satisfies the reduction constraint:

$$(\forall t \in V_{FS})(\forall s' \in S_{FS}).[t \in s' \Leftrightarrow ((\bigcup_{s \in g_{CHK}(s')} s) \cap k_{CHK}(t)) \neq \emptyset].$$

$\square$

Notice that the above reduction, combined with Theorem 45 and with the HE.IND-ID-CPA security of the hypergraph encryption scheme for Tree Access Hypergraphs described in Lemma 41, yields an efficient HE.IND-ID-CPA secure hypergraph encryption scheme HE_{FS} for the model of Forward-Security. This, in turn, can be easily shown to be equivalent to a scheme secure in the sense of forward-security against chosen-plaintext attacks (FSE.IND-ID-CPA) (*cf.* Chapter 2, Section 2.5).

Corollary 47. *A Forward-Secure Encryption scheme $\mathcal{E}_{\mathsf{FSE}}$ secure under FSE.IND-ID-CPA can be obtained from any HIBE, with a factor of $O(\log T)$ increase in storage complexity and communication complexity (T being the maximum number of instants.)*

Broadcast Encryption from Tree Access Hypergraphs

In virtue of Theorem 45 and by the existence of a HE.IND-ID-CPA secure hypergraph encryption scheme for any tree access hypergraph (*cf.* Lemma 41), to obtain a HE.IND-ID-CPA secure hypergraph encryption scheme HE_{BE} for the Broadcast Access Hypergraph $\mathcal{H}_{BE}$ it suffices to provide a reduction from $\mathcal{H}_{\mathcal{T}}$ to $\mathcal{H}_{BE}$, for some tree $\mathcal{T}$. Below we present such a reduction, based on the construction of [34] (*cf.* also Chapter 4, Section 4.5).

Lemma 48. *There exists a tree hierarchy $\mathcal{T}_{DF}$ such that $\mathcal{H}_{BE} \leq \mathcal{H}_{\mathcal{T}_{DF}}$, with $G = 2r-1$, $K = \frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$ and running time $Time(g_{DF}) = O(r \log N)$ (where r denotes the actual number of revoked users).*

Proof. Recall (Chapter 4, Section 4.5) that the construction of [34] extends the *Subset Difference* (SD) method of Naor *et al.* [65] to the public-key setting, by defining a hierarchy $\mathcal{T}'_{SD}$ whose leaves corresponds to the subsets of the SD cover family $\mathcal{S}_{SD}$. For notational convenience, below we denote this hierarchy by $\mathcal{T}_{DF}$ *i.e.*, $\mathcal{T}_{DF} = \mathcal{T}'_{SD}$.

As described in Definition 40, the hierarchy $\mathcal{T}_{DF}$ induces an access hypergraph $\mathcal{H}_{\mathcal{T}_{DF}} = (V_{DF}, S_{DF})$. We now define a reduction from $\mathcal{H}_{\mathcal{T}_{DF}}$ to $\mathcal{H}_{BE}$ (with $r_{max} = N$), describing the two functions $g_{DF} : S_{BE} \rightarrow \mathcal{P}(S_{DF})$ and $k_{DF} : V_{BE} \rightarrow \mathcal{P}(V_{DF})$. On input $s' \in S_{BE}$, g_{DF} first uses the cover algorithm of the SD method to partition s' into the subsets $\{S_{i_1 j_1}, \ldots, S_{i_k j_k}\}$.[2] Recall that each subset $S_{i_h j_h} (h = 1, \ldots, k)$, corresponds to a leaf l_h in the tree $\mathcal{T}_{DF}$: let p_h be the set of vertices in the path from the Root of $\mathcal{T}_{DF}$ to the leaf l_h corresponding to the set $S_{i_h j_h}$. Then, g_{DF} outputs the set $\{p_1, \ldots, p_k\} \subseteq S_{DF}$.

On input $v \in V_{BE}$ (v leaf in the tree $\mathcal{T}_{SD}$), k_{DF} first considers the path from the root of $\mathcal{T}_{SD}$ to v: let $v_0, v_1, \ldots, v_k \equiv v$ be all the ancestor of v, and denote by u_h the siblings of $v_h, h = 1, \ldots, k$. For $1 \leq i < j \leq k$ consider the subset difference S_{v_i, u_j} and the associated leaf $l_{v_i u_j}$ in $\mathcal{T}_{DF}$. Then, k_{DF} outputs the set $\{l_{v_i u_j} : 1 \leq i < j \leq k\}$.

It can be easily verified that the above construction satisfies the reduction constraint:

$$(\forall v \in V_{BE})(\forall s' \in S_{BE}).\ \left[v \in s' \Leftrightarrow \left(\left(\bigcup_{s \in g_{DF}(s')} s \right) \cap k_{DF}(v) \right) \neq \emptyset \right].$$

□

As argued above, Lemma 48 proves the existence of a HE.IND-ID-CPA secure hypergraph encryption scheme HE_{BE} for the Broadcast Encryption setting, which in turn induces a Broadcast Encryption scheme secure against chosen-plaintext attacks.

Corollary 49. *A Broadcast Encryption scheme $\mathcal{E}_{\mathsf{BE}}$ secure under* BE.*IND-ID-CPA can be obtained from any* HIBE*, with an increase of a factor of $O(\log^2 N)$ in storage complexity, of a factor of $2r$ for the communication complexity, and with encryption complexity increased by a factor of $2r$, plus $O(r \log N)$ time to compute g_{DF} (where r denotes the actual number of revoked users.)*

5.4.4 Product of Access Hypergraphs

Definition 50 (Product of hypergraphs).

Given two hypergraphs $\mathcal{H}_1 = (V_1, S_1)$ and $\mathcal{H}_2 = (V_2, S_2)$, define their product $\mathcal{H}_3 \doteq \mathcal{H}_1 \times \mathcal{H}_2$ to be the hypergraph whose vertices V_3 are the elements of the Cartesian

[2] The cover algorithm of [65] guarantees that $k < 2(N - |s'|)$.

product $V_1 \times V_2$, and whose hyperedges are the sets $s_1 \times s_2$, with $s_1 \in S_1$ and $s_2 \in S_2$ *i.e.*, $\mathcal{H}_3 = (V_3, S_3)$, with $V_3 \doteq V_1 \times V_2$ and $S_3 \doteq \{s_1 \times s_2 : s_1 \in S_1 \wedge s_2 \in S_2\}$.

The notion of product of hypergraphs provides a powerful way to combine the properties of two access hypergraphs into a single structure. In Theorem 51, we show that reductions between two pairs of access hypergraphs can be extended to a reduction between their products.

Theorem 51 (Multiplication Theorem)**.** *Let $\mathcal{H}_1, \mathcal{H}'_1, \mathcal{H}_2, \mathcal{H}'_2$ be access hypergraphs such that there exist reductions from $\mathcal{H}_1$ to $\mathcal{H}'_1$ and from $\mathcal{H}_2$ to $\mathcal{H}'_2$. Then there is also a reduction from $\mathcal{H}_3 = \mathcal{H}_1 \times \mathcal{H}_2$ to $\mathcal{H}'_3 = \mathcal{H}'_1 \times \mathcal{H}'_2$. In formula:*

$$((\mathcal{H}'_1 \leq \mathcal{H}_1) \wedge (\mathcal{H}'_2 \leq \mathcal{H}_2)) \Rightarrow ((\mathcal{H}'_1 \times \mathcal{H}'_2) \leq (\mathcal{H}_1 \times \mathcal{H}_2)).$$

Moreover, the resulting reduction has cover size $G_3 \doteq G_1 \cdot G_2$, key size $K_3 \doteq K_1 \cdot K_2$ and running time $Time(g_3) \doteq Time(g_1) + Time(g_2) + O(G_3)$.

Proof. By definition, $\mathcal{H}_3 = (V_3, S_3)$ where $V_3 = V_1 \times V_2$ and $S_3 = \{s_1 \times s_2 : s_1 \in S_1 \wedge s_2 \in S_2\}$; similarly, $\mathcal{H}'_3 = (V'_3, S'_3)$ where $V'_3 = V'_1 \times V'_2$ and $S'_3 = \{s'_1 \times s'_2 : s'_1 \in S'_1 \wedge s'_2 \in S'_2\}$.

To prove the theorem, we first define a reduction (g_3, k_3) from $\mathcal{H}_3$ to $\mathcal{H}'_3$ (where $g_3 : S'_3 \to \mathcal{P}(S_3), k_3 : V'_3 \to \mathcal{P}(V_3)$), given a reduction (g_1, k_1) from $\mathcal{H}_1$ to $\mathcal{H}'_1$ (where $g_1 : S'_1 \to \mathcal{P}(S_1), k_1 : V'_1 \to \mathcal{P}(V_1)$), and a reduction (g_2, k_2) from $\mathcal{H}_2$ to $\mathcal{H}'_2$ (where $g_2 : S'_2 \to \mathcal{P}(S_2), k_2 : V'_2 \to \mathcal{P}(V_2)$). For an arbitrary $s'_3 \in S'_3$ of the form $s'_3 = s'_1 \times s'_2$ with $s'_1 \in S'_1, s'_2 \in S'_2$ and for any $v'_3 \in V'_3$ of the form $v'_3 = (v'_1, v'_2)$ with $v'_1 \in V'_1, v'_2 \in V'_2$, define the two function g_3, k_3 as $g_3(s'_3) \doteq \{s_1 \times s_2 : s_1 \in g_1(s'_1) \wedge s_2 \in g_2(s'_2)\}$ and $k_3(v'_3) \doteq k_1(v'_1) \times k_2(v'_2)$.

To complete the proof, we need to show that the definition of (g_3, k_3) satisfies the reduction constraint:

$$(\forall v'_3 \in V'_3)(\forall s'_3 \in S'_3).[v'_3 \in s'_3 \Leftrightarrow ((\bigcup_{s_3 \in g_3(s'_3)} s_3) \cap k_3(v'_3)) \neq \emptyset]$$

Fix $v'_3 \in V'_3$ and $s'_3 \in S'_3$. First, note that

$$v'_3 \in s'_3 \Leftrightarrow (v'_1, v'_2) \in s'_1 \times s'_2 \Leftrightarrow v'_1 \in s'_1 \wedge v'_2 \in s'_2$$

Let

$$I_1 \doteq ((\bigcup_{s_1 \in g_1(s_1')} s_1) \cap k_1(v_1'))$$

$$I_2 \doteq ((\bigcup_{s_2 \in g_2(s_2')} s_2) \cap k_2(v_2'))$$

By definition of reduction from $\mathcal{H}_1$ to $\mathcal{H}_1'$ and from $\mathcal{H}_2$ to $\mathcal{H}_2'$, it holds that

$$v_1' \in s_1' \wedge v_2' \in s_2' \Leftrightarrow I_1 \neq \emptyset \wedge I_2 \neq \emptyset \Leftrightarrow \exists\, v_1 \in I_1 \wedge \exists\, v_2 \in I_2$$

By definition of I_1 and I_2, this can be rewritten as

$$(\exists\, v_1)(\exists\, v_2).[(v_1 \in k_1(v_1') \wedge v_1 \in \bigcup_{s_1 \in g_1(s_1')} s_1) \wedge (v_2 \in k_2(v_2') \wedge v_2 \in \bigcup_{s_2 \in g_2(s_2')} s_2)] \Leftrightarrow$$

$$(\exists\, v_1)(\exists\, v_2).[(v_1 \in k_1(v_1') \wedge v_2 \in k_2(v_2')) \wedge (v_1 \in \bigcup_{s_1 \in g_1(s_1')} s_1 \wedge v_2 \in \bigcup_{s_2 \in g_2(s_2')} s_2)]$$

Now, by construction we have

$$v_1 \in k_1(v_1') \wedge v_2 \in k_2(v_2') \Leftrightarrow v_3 = (v_1, v_2) \in k_3(v_3')$$

Moreover,

$$v_1 \in (\bigcup_{s_1 \in g_1(s_1')} s_1) \wedge v_2 \in (\bigcup_{s_2 \in g_2(s_2')} s_2) \Leftrightarrow$$

$$(\exists\, \bar{s}_1 \in g_1(s_1')).[v_1 \in \bar{s}_1] \wedge (\exists\, \bar{s}_2 \in g_2(s_2')).[v_2 \in \bar{s}_2] \Leftrightarrow$$

$$(\exists\, \bar{s}_1 \in g_1(s_1'))(\exists\, \bar{s}_2 \in g_2(s_2')).[(v_1, v_2) \in \bar{s}_1 \times \bar{s}_2 \Leftrightarrow$$

$$(\exists\, \bar{s}_3 \in g_3(s_3')).[(v_1, v_2) \in \bar{s}_3]] \Leftrightarrow$$

$$v_3 = (v_1, v_2) \in (\bigcup_{s_3 \in g_1(s_3')} s_3)$$

This completes the proof. □

5.4.5 Toward Forward-Secure Broadcast Encryption

Theorem 51, although simple, constitutes a crucial step towards the construction of a FSBE.IND-ID-CPA secure FSBE scheme: it enables a smooth and easy combination of the results of Lemma 46 and Lemma 48, thus yielding an efficient reduction to an access hypergraph which formalizes the setting of Forward-Secure Broadcast Encryption.

Definition 52 (Paired Trees Access Hypergraph).
Let $\mathcal{T}_1, \mathcal{T}_2$ be trees, and $\mathcal{H}_{\mathcal{T}_1}, \mathcal{H}_{\mathcal{T}_2}$ be the corresponding Tree Access Hypergraphs. The Paired Trees Access Hypergraph associated to $\mathcal{T}_1, \mathcal{T}_2$ is defined as $\mathcal{H}_{\mathcal{T}_1,\mathcal{T}_2} \doteq \mathcal{H}_{\mathcal{T}_1} \times \mathcal{H}_{\mathcal{T}_2}$.

Definition 53 (Forward-Secure Broadcast Access Hypergraph).
The access hypergraph for the Forward-Secure Broadcast model is:

$$\mathcal{H}_{FBE} \doteq \mathcal{H}_{FS} \times \mathcal{H}_{BE}.$$

It can be easily verified that a FSBE scheme secure in the sense of FSE.IND-ID-CPA is essentially equivalent to a HE.IND-ID-CPA secure hypergraph encryption $\mathsf{HE}_{\mathsf{FSBE}}$ for the Forward-Secure Broadcast Encryption model. Furthermore, by Theorem 51, Lemma 46 and Lemma 48, there exists an efficient reduction from $\mathcal{H}_{\mathcal{T}_{CHK},\mathcal{T}_{DF}}$ to $\mathcal{H}_{FBE}$; thus, by Theorem 45 and Lemma 41, we get the following:

Corollary 54. *A Forward-Secure Broadcast Encryption scheme can be obtained from any* HE*.IND-ID-CPA secure hypergraph encryption for Paired Trees Access Hypergraph, with an increase in storage complexity of a factor of* $O(\log T \log^2 N)$*, in communication complexity of a factor of* $2r$*, and with encryption complexity increased by a factor of* $2r$*, plus* $O(\log T + r \log N)$ *time to compute* g_{FBE} *(where* T *is the maximum number of instants,* N *is size of the users universe and* r *is the actual number of revoked users.)*

One question that arises naturally is whether a hypergraph encryption for Paired Trees Access Hypergraphs can be obtained combining, in a black-box fashion, hypergraph encryptions for Tree Access Hypergraphs; or, more generally, how to securely and efficiently combine hypergraph encryptions $\mathsf{HE}_1, \mathsf{HE}_2$ for $\mathcal{H}_1, \mathcal{H}_2$ into a hypergraph encryption HE_3 for $\mathcal{H}_3 \doteq \mathcal{H}_1 \times \mathcal{H}_2$. Such a generic combination would essentially need to define the secret key of an entity $(v_1, v_2) \in \mathcal{H}_3$ as the juxtaposition of the secret keys of $v_1 \in \mathcal{H}_1$ and the secret keys of $v_2 \in \mathcal{H}_2$ *i.e.*, $\mathsf{HE}_3.\mathsf{SK}_{(v_1,v_2)} \doteq (\mathsf{HE}_1.\mathsf{SK}_{v_1}, \mathsf{HE}_2.\mathsf{SK}_{v_2})$.

Unfortunately, this way of combining secret keys is not secure, as demonstrated by the following generic attack. Suppose $\mathcal{H}_1$ contains vertices v_1, v_1' and hyperedges s_1, s_1' such that $s_1 \subsetneq s_1'$ and $v_1 \in s_1, v_1' \in s_1' \setminus s_1$; and similarly, $\mathcal{H}_2$ contains vertices v_2, v_2' and hyperedges s_2, s_2' such that $s_2 \subsetneq s_2'$ and $v_2 \in s_2, v_2' \in s_2' \setminus s_2$. Then, it would be possible for an adversary holding the secret keys of vertices $v'_{\text{bad}} \doteq (v_1, v_2')$ and $v''_{\text{bad}} \doteq (v_1', v_2)$, to recover the secret key of $(v_1, v_2) \in s_1 \times s_2$, even though neither v'_{bad} nor v''_{bad} belongs to that hyperedge, which violates security for HE_3 without breaking neither HE_1 nor HE_2.

Motivated by the lack of such a generic construction, in [82] we introduced the concept of *Paired Hierarchical Identity Based Encryption* (PHIBE), an extension of HIBE that we discussed in Chapter 2, Section 2.4.3. In the next section, we argue that an (efficient) HE.IND-ID-CPA secure access hypergraph for Paired Trees Access Hypergraphs can be obtained from any (efficient) PHIBE.IND-ID-CPA secure PHIBE scheme (Lemma 55).

5.4.6 PHIBE as Hypergraph Encryption for Paired Access Hypergraphs

Lemma 55. *Let $\mathcal{T}^\ell, \mathcal{T}^r$ be trees and let $\mathcal{H}_{\mathcal{T}^\ell,\mathcal{T}^r}$ be the corresponding Paired Trees Access Hypergraph. A* HE.*IND-ID-CPA secure hypergraph encryption scheme for $\mathcal{H}_{\mathcal{T}^\ell,\mathcal{T}^r}$ can be obtained from any* HIBE.*IND-ID-CPA secure* PHIBE *scheme (with essentially the same efficiency parameters of the corresponding* PHIBE *scheme).*

Proof. Consider the following construction of a hypergraph encryption scheme HE for $\mathcal{H}_{\mathcal{T}^\ell,\mathcal{T}^r}$ based on any PHIBE scheme. Let $s = s^\ell \times s^r$ and $v = (v^\ell, v^r)$. The HE.Extract algorithm generates $\mathsf{params}_{\mathsf{HE}}$ and $\mathsf{master}_{\mathsf{HE}}$ as follows: $\mathsf{params}_{\mathsf{HE}} \doteq \mathsf{params}_{\mathsf{PHIBE}}$ and $\mathsf{HE.SK}_{(v^\ell,v^r)} \doteq \mathsf{PHIBE.Extract}^*(\mathsf{params}_{\mathsf{PHIBE}}, \mathrm{PID}_v, \mathsf{master}_{\mathsf{PHIBE}})$ *i.e.*, each entity $v = (v^\ell, v^r)$ receives the secret key corresponding to its paired hierarchical identifier PID_v, and the star in $\mathsf{PHIBE.Extract}^*$ denotes the fact that the PHIBE.Extract algorithm may have to be run multiple times, depending on the levels at which the left and right components of the paired hierarchical identifier PID_v appears in $\mathcal{T}^\ell$ and $\mathcal{T}^r$, respectively. Notice that this construction allows for a compact representation of $\mathsf{master}_{\mathsf{HE}}$ which consists of $\mathsf{master}_{\mathsf{PHIBE}}$.

As for the encryption and decryption algorithms, $\mathsf{HE.Encrypt}$ and $\mathsf{HE.Decrypt}$, they can be obtained from the corresponding components of PHIBE:

$$\mathsf{HE.Encrypt}(\mathsf{params}_{\mathsf{HE}}, s, m) \doteq \mathsf{PHIBE.Encrypt}(\mathsf{params}_{\mathsf{PHIBE}}, \mathrm{PID}_{\hat{v}}, m)$$
$$\mathsf{HE.Decrypt}(\mathsf{HE.SK}_v, s, C) \doteq \mathsf{PHIBE.Decrypt}(\mathsf{params}_{\mathsf{PHIBE}}, \mathsf{PHIBE.SK}_{\mathrm{PID}_{\hat{v}}}, C)$$

where: $\hat{v} = (\hat{v}^\ell, \hat{v}^r)$, $\hat{v}^\ell$ is the minimal element in the set s^ℓ (w.r.t. the $\preceq^\ell$ relationship in $\mathcal{T}^\ell$) and $\hat{v}^r$ is the minimal element in the set s^r (w.r.t. the $\preceq^r$ relationship in $\mathcal{T}^r$); $\mathrm{PID}_{\hat{v}}$ denotes the paired hierarchical identifier associated to $\hat{v}$; and finally, $\mathsf{PHIBE.SK}_{\mathrm{PID}_{\hat{v}}}$ is the secret key corresponding to the paired hierarchical identifier $\mathrm{PID}_{\hat{v}}$, which entity v either has (if $v^\ell = \hat{v}^\ell$ and $v^r = \hat{v}^r$), or can derive from its secret information using the $\mathsf{PHIBE.Extract}$ algorithm.

It can be easily verified that applying the above transformation to a secure PHIBE scheme yields a hypergraph encryption for the Tree Access Hypergraph $\mathcal{H}_{\mathcal{T}^\ell,\mathcal{T}^r}$ corresponding to the given hierarchy $\mathcal{T}^\ell \times \mathcal{T}^r$. □

Remark 56. Using the specific construction for PHIBE resulting from the work of [16], we get an hypergraph encryption for Paired Trees Access Hypergraphs under the decisonal bilinear Diffie-Hellman inversion (DBDHI) assumption with *communication complexity* proportional to d_1 and *storage* and *encryption complexity* proportional to $d_1 + d_2$, where d_1 and d_2 denote the depths of the trees $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.

Remark 57. Reasoning along the same lines of Section 5.4.1, we notice that the full power of PHIBE (which allows both hierarchies to grow dynamically and independently) is not strictly necessary in Lemma 55 above, since the hierarchies defined by $\mathcal{T}^\ell$ and $\mathcal{T}^r$ in $\mathcal{H}_{\mathcal{T}^\ell,\mathcal{T}^r}$ are fixed. Thus, we could as well have based our construction on *Paired Tree Encryption*, a generalization of the notion of *Binary Tree Encryption* (BTE, [26]) to the case of pairs of trees, each with polynomial (in the security parameter λ) breadth and depth.

5.5 Putting it all Together

5.5.1 FBE from the Access Hypergraph Framework

Having obtained all the main ingredients from the Access Hypergraph Framework, our main result for this chapter is now a straightforward consequence of Corollary 54, Lemma 55 and Remark 56:

Theorem 58. *Under the decisional bilinear Diffie-Hellman inversion (*DBDHI*) assumption, there exists an efficient and secure Forward-Secure Public-Key Broadcast Encryption scheme with storage complexity* $O(\log T \log^2 N(\log T + \log N))$*, encryption complexity* $O(r(\log T + \log N))$ *and communication complexity* $O(r \log T)$*.*

5.5.2 An Alternative Formulation based on fs-HIBE

An alternative way to describe our approach to obtain FSBE scheme is by employing a *Forward-Secure Hierarchical Identity-Based Encryption scheme* fs-HIBE over the construction of the public-key broadcast encryption that we described in Chapter 4, Section 4.5.

Recall that such construction is based on a labeled tree $\mathcal{T}'_{SD}$, whose leaves correspond to all the subsets S_{ij} in the cover family $\mathcal{S}_{SD}$ for the SD symmetric-key broadcast encryption scheme of [65] (*cf.* Chapter 3, Section 3.3). Moreover, vertices in $\mathcal{T}'_{SD}$ are associated to a hierarchical identifier by a mapping HID$(\cdot)$.

However, to get forward-security, each vertex w in $\mathcal{T}'_{SD}$ is no longer associated to a *single* private key; rather, for each time period t, there is a secret key $\mathsf{SK}_{t,\mathrm{HID}(w)}$, which can be computed using the Extract/Update algorithms of fs-HIBE.

Construction:

- FSBE.Setup(k, r_{max}, T, N): Run algorithm fs-HIBE.Setup(k, T, N), set $\mathsf{params}_{\mathsf{FSBE}} \doteq (\mathsf{params}_{\mathrm{fs}}\text{-}\mathsf{HIBE}, \mathrm{HID}(\cdot))$ and $\mathsf{master}_{\mathsf{FSBE},0} \doteq \mathsf{master}_{\text{fs-}\mathsf{HIBE},0}$.

- FSBE.Register$(\mathsf{master}_{\mathsf{FSBE},t}, u, t)$: Consider the path from the root of $\mathcal{T}_{SD}$ to the leaf representing user u; let $u_0, u_1, \ldots, u_n \equiv u$ be all the ancestors of u, and denote by v_h the sibling of $u_h, h = 1, \ldots, n$. For $1 \leq i < j \leq n$ consider the subset difference S_{u_i,v_j} and the associated leaf l_{u_i,v_j} in $\mathcal{T}'_{SD}$; let w_{u_i,v_j} be the parent of l_{u_i,v_j} in $\mathcal{T}'_{SD}$. Then, starting from $\mathsf{master}_{\mathsf{PHIBE}}$, recursively apply the $\mathsf{Extract}$ algorithm of fs-HIBE to derive the secret key $\mathsf{SK}_{t,\mathrm{HID}(w_{u_i,v_j})}$. Set the user's secret key $\mathsf{FSBE.SK}_{t,u} \doteq \{\langle \mathrm{HID}(w_{u_i,v_j}), \mathsf{SK}_{t,\mathrm{HID}(w_{u_i,v_j})}\rangle \mid 1 \leq i < j \leq n\}$.

- $\mathsf{FSBE.Update}(PK, u, t, \mathsf{FSBE.SK}_{t,u})$: For each $1 \leq i < j \leq n$ such that $\langle \mathrm{HID}(w_{u_i,v_j}), SK_{t,\mathrm{HID}(w_{u_i,v_j})}\rangle \in \mathsf{FSBE.SK}_{t,u}$, run $\mathsf{PHIBE.Update}(t, \mathrm{HID}(w_{u_i,v_j}), \mathsf{SK}_{t,\mathrm{HID}(w_{u_i,v_j})})$ to compute $\mathsf{SK}_{t+1,\mathrm{HID}(w_{u_i,v_j})}$ and erase $\mathsf{SK}_{t,\mathrm{HID}(w_{u_i,v_j})}$. Finally, return $\mathsf{FSBE.SK}_{t+1,u} \doteq \{\langle \mathrm{HID}(w_{u_i,v_j}),\ \mathsf{SK}_{t+1,\mathrm{HID}(w_{u_i,v_j})}\rangle \mid 1 \leq i < j \leq n\}$. For the Center, call the fs-$\mathsf{HIBE.Update}(t, \mathsf{master}_{\text{fs-HIBE},t})$ to compute the master secret key $\mathsf{master}_{\mathsf{FSBE},t+1} = \mathsf{master}_{\text{fs-HIBE},t+1}$ for time period $t + 1$.

- $\mathsf{FSBE.Encrypt}(PK, m, t, \mathcal{R})$: First run the *Cover* algorithm (cf. Naor *et al.* [65]) to partition the set $\mathcal{N} \setminus \mathcal{R}$ into the subsets $\{S_{i_1,j_1}, \ldots, S_{i_z,j_z}\}$. (The *Cover* algorithm guarantees that $z < 2|\mathcal{R}|$.) For each $h = 1, \ldots, z$, run algorithm fs-$\mathsf{HIBE.Encrypt}(\mathsf{params}_{\mathsf{PHIBE}}, t, \mathrm{HID}(w_{u_{i_h},v_{j_h}}), m)$ to compute ciphertext C_h. Return $C \doteq \langle \mathrm{HID}(l_{u_{i_1},v_{j_1}}), \ldots, \mathrm{HID}(l_{u_{i_z},v_{j_z}}), C_1, \ldots, C_z\rangle$.

- $\mathsf{FSBE.Decrypt}(PK, u, t, \mathsf{FSBE.SK}_{t,u}, C)$: By definition of the $\mathsf{Register}$ algorithm, if the ciphertext $C \doteq \langle \mathrm{HID}(l_{u_{i_1},v_{j_1}}), \ldots, \mathrm{HID}(l_{u_{i_z},v_{j_z}}), C_1, \ldots, C_z\rangle$ was constructed for a subset $\mathcal{R}$ of revoked users which does not include u, then among the hierarchical identifiers included in C, there exists one (say, $\mathrm{HID}(l_{u_{i_h},v_{j_h}})$) which is a descendent of one of the ID-tuples (say, $\mathrm{HID}(w_{u_i,u_j})$) for which user u received the corresponding secret key when he joined the system. Let $\mathsf{SK}_{t,\mathrm{HID}(w_{u_i,u_j})}$ be the value of such a secret key at time period t. Starting from such key, recursively apply fs-$\mathsf{HIBE.Extract}$ to derive the secret key $\mathsf{SK}_{t,\mathrm{HID}(l_{u_{i_h},v_{j_h}})}$ corresponding to $\mathrm{HID}(l_{u_{i_h},v_{j_h}})$. Then, run algorithm fs-$\mathsf{HIBE.Decrypt}(\mathsf{params}_{\mathsf{PHIBE}}, t, \mathrm{HID}_h,\ \mathsf{SK}_{t,\mathrm{HID}(l_{u_{i_h},v_{j_h}})}, C_h)$ to obtain the corresponding message m.

Remark 59. Using the specific construction for fs-HIBE presented in [16], this yields a Forward-Secure Broadcast Encryption Scheme, with *storage complexity* $O(\log^2 N(\log T + \log N))$, with *encryption complexity* $O(r(\log T + \log N))$ and *communication complexity* $O(r \log T)$.

5.6 Achieving Chosen-Ciphertext Security

For the sake of clarity, in Section 5.3, Section 5.4, and Section 5.4.6 we focused on the basic notion of IND-ID-CPA security, providing definitions, generic results and an efficient construction specific for that level of security. Realistic usage scenarios, however, call for a higher level of security *i.e.*, IND-ID-CCA security.

For most of the results of Section 5.4, replacing IND-ID-CPA secure components with their IND-ID-CCA secure counterparts suffices to yield similar results for the IND-ID-CCA security case. In particular, this is true for Lemma 41 (and for Corollary 47 and Corollary 49 which directly follow from it), as well as for Lemma 55 (and Corollary 54).

These simple syntactic changes, however, are not sufficient to extend Theorem 45 (the Reduction Theorem) to the IND-ID-CCA setting, since the underlining argument is based on multiple encryptions, which can be plainly combined without incurring in loss of security only for the IND-ID-CPA case. To deal with IND-ID-CCA security, we leverage the result of Dodis and Katz [39] which shows how to securely combine multiple chosen-ciphertext encryptions, without the use of the Random Oracle Methodology.

More precisely, in securely broadcasting multiple encryptions of a message m (as mandated by the Reduction Theorem), the Center first generates the pair $(\mathsf{params}_{sig}, \mathsf{master}_{sig})$ for a one-time signature scheme, and then computes $e_1 \leftarrow Enc_1(m, \mathsf{params}_{sig}), \ldots, e_k \leftarrow Enc_k(m, \mathsf{params}_{sig})$. Finally, the Center generates $\sigma = Sig(e_1, \ldots, e_k)$ and broadcast the ciphertext $\langle \mathsf{params}_{sig}, e_1, \ldots, e_k, \sigma \rangle$.

Using this technique, we can prove a Reduction Theorem (similar to Theorem 45) for the IND-ID-CCA setting, with a loss in the efficiency parameters of a constant factor due to the use of signatures in the model of [39].

Chapter 6

Traitor Tracing with Optimal Transmission Rate

6.1 Introduction

As pointed out by Kiayias and Yung in [57], an important problem in designing practical traitor tracing schemes is to ensure a low *transmission rate* (defined as the asymptotic ratio of the size of ciphertexts over the size of plaintexts), while at the same time minimize the *secret-* and the *public-storage* rates (similarly defined as the asymptotic ratio of the size of user-keys and of public-keys over the size of plaintexts).[1]

Under this terminology, the transmission rate of all the above mentioned solutions is linear w.r.t. the maximal number t of traitors, whereas in [57], Kiayias and Yung show that if the plaintexts to be distributed are large (which is already the case for most applications of traitor tracing, such as distribution of multimedia content), then it is possible to obtain ciphertexts with constant expansion rate. Their solution is

[1]We are adopting here a terminology slightly different from the one of [57], which uses the term *ciphertext/user-key/public-key* rates, for what we called *transmission/secret-storage/public-storage* rates. Moreover, in [57] *transmission rate* refers to the sum of the all the three rates. Our choice is of course mostly a matter of taste: we prefer the terminology of this paper as it makes more evident the role played by each quantity in a concrete implementation of the system.

based on collusion-secure fingerprint codes [23, 79] and its parameters are summarized in Figure 6.1.

Besides the clear benefit in terms of communication-efficiency, schemes with constant transmission rate also enjoy efficient black-box traceability, while schemes with linear transmission rate are inherently more limited in this regard [55] (*e.g.,* the black-box traitor tracing of [17] takes time proportional to $\binom{n}{t}$).

An extension to the work of [57] was recently proposed in [27], which further introduced the notion of *(local) public traceability*: Whereas in traditional traitor tracing schemes only the security manager could execute the tracing procedure (thanks to the knowledge of some piece of information whose secrecy is crucial for the overall security of the system), in a scheme with public traceability every one can run the tracing algorithm (or at least its preliminary, interactive part which requires the availability of the pirate decoder, in which case one talks of local public traceability). Figure 6.1 also summarizes the main characteristics of the scheme in [27].

One could think that existing traitor tracing schemes with linear transmission rate (*e.g.* [17]) could be easily turned into schemes with constant transmission rate by means of hybrid encryption: To send a large message, pick a random session key, encrypt it using the given traitor tracing scheme, and append a symmetric-key encryption of the message under the chosen session key. The problem with this approach is that it opens the way to a simple pirate strategy: Just decrypt the session key and make it available to the "customers" on the black market. Since all the users recover the same session key, even if the security manager got hold of the leaked session key, it would have no way to trace it back to the source of the leakage. Although it may be possible to augment this idea (*e.g.* using dynamic traitor tracing), it is not at all clear that one would obtain a scheme with optimal rate and efficient black-box traceability, as we do in our scheme.

Notice that such "re-broadcasting" strategy does not apply when the traitor tracing scheme is used to directly encrypt the content, for that would likely pose too high a demand on the up-bandwidth available to the pirate.

	Transmission Rate	SK-Storage Rate	PK-Storage Rate	BB Tracing	Local Public Traceability
[17]	$2t+1$	$2t$	$2t+1$	$\times$	$\times$
[57]	3	2	4	$\surd^*$	$\times$
[27]	1	2	1	$\times$	$\times$
Repaired [27]	3	2	6	$\surd$	$\surd$
Our Scheme	1	2	11	$\surd$	$\surd$

Figure 6.1: Comparison of rates (*transmission*, *secret-* and *public-storage* rates) and tracing features (*black-box tracing* and *local public traceability*) between existing schemes and our construction.
The "*" in the table refers to the fact that the scheme of [57] can support black-box tracing with the tracing algorithm described in Section 6.6.2. The row labeled "Repaired [27]" refers to the variant of the scheme of [27] modified to support black-box tracing.

6.2 Our Results

In this chapter, based on our work in [41], we present the first Public-Key Traitor Tracing scheme with efficient black-box traitor tracing and local public traceability in which the transmission rate is asymptotically 1, which is optimal. Encryption involves the same amount of computation as in [27], while decryption is twice as fast. Figure 6.1 summarizes the comparison of our construction with existing schemes, in terms of both rates and tracing features. We remark that no scheme with optimal transmission rate was known, even in the symmetric-key setting.

As additional contribution, we point out and resolve an issue in the black-box traitor tracing mechanism in the scheme of [57], which was claimed to work with just a single query to the pirate decoder. We show that this is not the case, and fix the problem by presenting a black-box tracing mechanism that employs a number of black-box queries proportional to the length of the plaintext, while not requiring any change to the scheme of [57].

We also show that [27], which extends [57] and inherits its tracing mechanism, inherits in fact the above-mentioned problem, too. In this case, however, the consequences are more severe. First, since the notion of local public traceability is only meaningful in the black-box setting, this in particular means that, as given, [27] *does not* possess any public traceability features. Second, fixing the black-box functionality and the local public traceability of [27] requires substantial changes to the scheme, which intrinsically conflict with the optimizations put up by [27] to achieve optimal transmission rate. In other words, [27] can either provide optimal transmission rate with only non-black-box tracing and no public traceability features, or support local public traceability with sub-optimal transmission rate, but cannot achieve both at the same time.

On the contrary, our construction *simultaneously* supports (local) public traceability, black-box traitor tracing (with the same number of queries as in the repaired [57]-scheme), and optimal transmission rate, all under the decisional bilinear Diffie-Hellman (DBDH) assumption in the standard model.

Message and Keys Lengths in a Concrete Instantiation. Existing constructions of traitor tracing schemes with constant transmission rate (including ours) are based on the use of collusion-secure fingerprint codes [23, 79], and in particular are applicable for messages of size proportional to the length of the code, which in the case of the optimal codes due to Tardos [79] is $O(t^2(\log n + \log \frac{1}{\varepsilon}))$ (where the hidden constant is less than 100). For a typical choice of the parameters, *e.g.* user population $n = 2^{30}$, tracing error probability $\varepsilon = 2^{-30}$ and traceable threshold $t = 30$,[2] the length of each codeword is about 5 million bits. Instantiating our construction with codes of such length, yields a scheme with plaintext and ciphertext of size 41 MBytes. (The ciphertext size is equal to the plaintext size, as the additive overhead is less than 1 KByte.) These values are well within the range of multimedia applications, since 41 MBytes roughly corresponds to 33 seconds of DVD-quality (high-resolution) video, 4 minutes of VCD-quality (lowresolution) video and 25–50 minutes of audio.

As for the key size, secret keys require roughly 206 MBytes, whereas the public key takes 1.8GByte. Although quite large, such public key could be stored in commodity hardware (*e.g.,* it would fit in the hard disk of the iPod), whereas the user secret key could be kept in a Secure Digital, like those commonly available for PDAs.

Organization. Section 6.4 and Section 6.5 review the traitor tracing schemes of [57] and [27]. In Section 6.6, we point out a flaw in the tracing algorithms of [57] and [27] and propose fixes. We present our new traitor tracing scheme and its security analysis in Section 6.7.

6.3 Formal Model

6.3.1 TT: Syntax

Definition 60 (Public-Key Traitor Tracing Scheme). *A public-key traitor tracing scheme* $\mathcal{E}_{\mathsf{TT}}$ *is a 5-tuple of probabilistic polynomial-time algorithms (*Setup*,* Register*,* Encrypt*,* Decrypt*,* Trace*), where:*

- Setup *takes as input a security parameter* 1^κ*, a collusion threshold* $1^{t_{\max}}$ *and the total number* N *of users in the system. It returns the public key* $\mathsf{params}_{\mathsf{TT}}$ *for the scheme, along with some master secret key* $\mathsf{master}_{\mathsf{TT}}$ *needed to generate user keys and to trace traitors from pirate decoder (*cf. Register *and* Trace*).*

- Register *takes as input the secret information* $\mathsf{master}_{\mathsf{TT}}$*, the identity* u *of the new user.* Register *outputs a "fingerprinted" user key* SK_u *good for decryption with* Decrypt.[3]

[2]A content distribution network afflicted by pirate coalitions of size larger than few dozens, has arguably bigger issues to solve, than finding out the identity of the traitors.

[3]Equivalently, we can think of Setup as outputting a vector of user keys, one per each user in the system; we will refer to either formalization interchangeably.

- $\mathsf{Encrypt}$ *takes as input the public key* $\mathsf{params}_{\mathsf{TT}}$*, a message* m *in the message space* $\mathcal{M}$*, and returns a (randomized) ciphertext* ψ*.*

- $\mathsf{Decrypt}$ *takes as input the public key* $\mathsf{params}_{\mathsf{TT}}$*, the identity* u *of a user, the user's secret key* SK_u *and a ciphertext* ψ*.* $\mathsf{Decrypt}$ *returns the message* m *or the special rejection symbol* $\perp$*.*

- Trace *takes as input the master secret key* $\mathsf{master}_{\mathsf{TT}}$*, the public key* $\mathsf{params}_{\mathsf{TT}}$*, and black-box access to a "pirate" decoder capable of inverting the* $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, \cdot)$ *functionality.* Trace *returns the identity of one of the traitors that contributed his/her user key for the realization of the pirate decoder, or* $\emptyset$ *upon failure.*

A traitor tracing scheme $\mathcal{E}_{\mathsf{TT}}$ should satisfy the following correctness constraint: for any pair $(\mathsf{params}_{\mathsf{TT}}, \mathsf{params}_{\mathsf{TT}})$ output by $\mathsf{TT.Setup}(1^\lambda, 1^{t_{\max}}, N)$, any user u with secret key SK_u (properly generated for user u) and any session message m:

$$m = \mathsf{TT.Decrypt}(\mathsf{params}_{\mathsf{TT}}, u, \mathsf{SK}_u, \mathsf{TT.Encrypt}(\mathsf{params}_{\mathsf{TT}}, m)).$$

Definition 61 (Public Traceability and Local Public Traceability [27])**.** *A public-key Traitor Tracing scheme is said to support: 1)* public traceability *if the* Trace *algorithm can be implemented without the master secret key* $\mathsf{master}_{\mathsf{TT}}$*; or 2)* local public traceability *if the* Trace *algorithm can be split in an on-line phase, in which the pirate decoder can be queried without knowledge of the secret key, and an off-line phase, without access to the pirate decoder, that can retrieve the identity of the traitor from the master secret key and the output of the publicly executable on-line phase.*

6.3.2 TT: Security

Requirements on the Encryption Functionality.

For security, encryption of distinct messages under a traitor tracing scheme should look indistinguishable to any efficient algorithm that is allowed to pick the two messages based on the public key of the system, but without knowledge of any user key:

Definition 62 (Indistinguishability under Chosen-Plaintext Attack)**.** *A public-key traitor tracing scheme satisfies* $\varepsilon_{\mathrm{ind}}$*-indistinguishability if, for any pair of probabilistic polynomial-time algorithms* (A_1, A_2)*:*

$$Pr\left[A_2(\tau, \psi^*) = b^* \left| \begin{array}{c} (\mathsf{params}_{\mathsf{TT}}, \mathsf{master}_{\mathsf{TT}}) \overset{R}{\leftarrow} \mathsf{Setup}(1^\kappa, 1^{t_{\max}}, N), \\ (m_0, m_1, \tau) \overset{R}{\leftarrow} A_1(\mathsf{params}_{\mathsf{TT}}), \\ b^* \overset{R}{\leftarrow} \{0,1\}, \psi^* \overset{R}{\leftarrow} \mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, m_{b^*})] \end{array} \right. \right] \leq \frac{1}{2} + \varepsilon_{\mathrm{ind}},$$

where the probability is over b^**, and the random coins of* A_1*,* A_2*,* Setup*, and* $\mathsf{Encrypt}$*.*

Requirements on the Tracing Functionality.

For tracing, the security manager ought to be able to "extract" the identity of at least one of the traitors, from any efficient pirate decoder[4] that can be (efficiently) produced given access to up to t adaptively chosen user keys:

Definition 63. *A public-key traitor tracing scheme is* $\varepsilon_{\mathrm{trac}}$*-traceable if for any probabilistic polynomial-time algorithm* $\mathcal{A}$*, it holds that:*

$$Pr\left[\mathsf{Trace}^D(\mathsf{params}_{\mathsf{TT}}, \mathsf{master}_{\mathsf{TT}}) \in T \left| \begin{array}{c} (\mathsf{params}_{\mathsf{TT}}, \mathsf{master}_{\mathsf{TT}}) \overset{R}{\leftarrow} \mathsf{Setup}(1^\kappa, 1^{t_{\max}}, N), \\ D \overset{R}{\leftarrow} A(\mathsf{params}_{\mathsf{TT}})^{\mathsf{Register}(\mathsf{master}_{\mathsf{TT}}, \cdot)} \end{array} \right. \right]$$

$$\geq 1 - \varepsilon_{\mathrm{trac}}$$

where T *is the set of up to* $t_{\max}$ *indices on which* $\mathcal{A}$ *queried the* $\mathsf{Register}(\mathsf{master}_{\mathsf{TT}}, \cdot)$ *oracle,* D *successfully inverts* $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, \cdot)$ *(almost) everywhere (in probabilistic polynomial-time), and the probability is over the random choices of* Setup*,* $\mathsf{Register}$*,* $\mathcal{A}$*,* D *and* Trace*.*

6.4 The KY Public-Key Traitor Tracing Scheme

We first briefly review the scheme of Kiayias and Yung [57]. Their approach consists of first devising a two-user traitor tracing withstanding just a single traitor, and then

[4]Following [66], we assume that it is sensible for the security manager to focus only on pirate decoders that work with probability very close to 1.

extend it to the multi-user setting using collusion-secure codes.

6.4.1 The Two-User Sub-Scheme

Setup: Given a security parameter 1^κ, the algorithm works as follows:

Step 1: Generate a κ-bit prime q and a group $\mathbb{G}$ of order q in which the DDH problem is difficult. Let P be a generator of $\mathbb{G}$.[5]

Step 2: Pick random elements $a, c \in \mathbb{Z}_q^*$, and set $Q \doteq aP$, $Z \doteq cP$. The private key of the security manager is set to be the pair $\mathsf{master}_{\mathsf{TT}} \doteq (a, c)$.

Step 3: Choose a universal hash function $H : \mathbb{G}_1 \to \{0,1\}^\kappa$, and set the public key as $\mathsf{params}_{\mathsf{TT}} \doteq (q,\, \mathbb{G},\, H,\, P,\, Q,\, Z)$. The message space is $\mathcal{M} \doteq \{0,1\}^\kappa$.

Register: The security manager selects two linearly independent vectors (α_0, β_0), $(\alpha_1, \beta_1) \in \mathbb{Z}_q^2$ such that $\alpha_\sigma + a\beta_\sigma = c \bmod q$, for $\sigma \in \{0,1\}$. This implies: $Z = cP = \alpha_\sigma P + \beta_\sigma Q$, for $\sigma \in \{0,1\}$. The secret key of user u_σ is then set to be $\mathsf{SK}_\sigma \doteq (\alpha_\sigma, \beta_\sigma)$, for $\sigma \in \{0,1\}$.

Encrypt: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can encrypt a message $m \in \mathcal{M}$ by first selecting a random $k \in \mathbb{Z}_q$ and then creating the ciphertext $\psi \doteq \langle A, B, C \rangle \in \mathbb{G}^2 \times \mathcal{M}$ where

$$A \doteq kP, \quad B \doteq kQ, \quad C \doteq m \oplus H(kZ).$$

Decrypt: Given a ciphertext $\psi = \langle A, B, C \rangle \in \mathbb{G}^2 \times \mathcal{M}$, user u_σ computes $kZ = \alpha_\sigma A + \beta_\sigma B$ and recovers $m = C \oplus H(kZ)$.

Trace: To trace a decoder D back to the identity of the traitor, the security manager picks two distinct random values $k, k' \in \mathbb{Z}_q$, along with a random $\hat{m} \in \mathcal{M}$, and feeds D with the "illegal" ciphertext $\hat{\psi} \doteq \langle k'P, kQ, \hat{m} \rangle$. If the output of D is $\hat{m} \oplus H(k'\alpha_\sigma P + k\beta_\sigma Q)$, then the algorithm returns the identity u_σ as the traitor; otherwise it outputs $\emptyset$.

[5]Even though [57] used the multiplicative notation, we use here the additive notation for the sake of consistency with the rest of the paper (*cf.* Footnote 6).

In [57], the authors show that the above two-user scheme is secure and traceable (for up to 1 traitor) in the sense of Definitions 62 and 63 under the DDH assumption (*cf.* Chapter 2, Section 2.3).

6.4.2 The Multi-User Scheme

Let $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\}$ be an (ε, t, n, v)-collusion-secure code over the alphabet $\{0, 1\}$ (*cf.* Chapter 2, Definition 7).

At a high level, the multi-user scheme of [57] is obtained by concatenating in parallel v instantiations of the two-user scheme from Section 6.4.1, resilient against a single traitor. Decryption keys for the multi-user scheme are then obtained by concatenating the keys for the v two-user sub-schemes, according to the codewords of $\mathcal{C}$: in other words, user u_i (associated to the codeword $\omega^{(i)}$) is given key $(K_{1,\omega_1^{(i)}}, ..., K_{v,\omega_v^{(i)}})$, where $\omega_j^{(i)}$ is the j-th bit of the codeword $\omega^{(i)}$, and $K_{j,0}, K_{j,1}$ are the keys for the j-th instantiation of the basic two-user sub-scheme.

Setup: Given security parameters 1^κ, 1^t and ε, the algorithm works as follows:

Step 1: Generate a κ-bit prime q and a group $\mathbb{G}$ in which the DDH problem is difficult.[6] Generate an (ε, t, n, v)-collusion-secure code $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\}$ over $\{0, 1\}$.

Step 2: For each $j = 1, \ldots, v$, let P_j be a generator of $\mathbb{G}$, pick random $a_j, c_j \in \mathbb{Z}_q^*$, and set $Q_j \doteq a_j P_j$, $Z_j \doteq c_j P_j$. For each $j = 1, \ldots, v$, compute two linearly independent vectors $(\alpha_{j,0}, \beta_{j,0})$, $(\alpha_{j,1}, \beta_{j,1})$ in $\mathbb{Z}q^2$ such that $\alpha_{j,\sigma} + a\beta_{j,\sigma} = c_j \bmod q$, for $\sigma \in \{0, 1\}$. The private key of the security manager is set to be $\mathsf{master_{TT}} \doteq (a_j, \alpha_{j,0}, \beta_{j,0}, \alpha_{j,1}, \beta_{j,1})_{j=1,\ldots,v}$.

[6]Even though [57] used the multiplicative notation, we use here the additive notation for the sake of consistency with the rest of the paper. Notice, however, that $\mathbb{G}$ should not be identified with the group $\mathbb{G}_1$ used elsewhere in this paper, and in particular $\mathbb{G}$ should not be equipped with a bilinear map, for that would violate the required hardness of the DDH problem in $\mathbb{G}$.

Step 3: Choose a universal hash function $H : \mathbb{G} \rightarrow \{0,1\}^{\kappa}$, and set the public key to $\mathsf{params}_{\mathsf{TT}} \doteq (q,\ \mathbb{G},\ H,\ (P_1,\ Q_1,\ Z_1),\ \ldots,\ (P_v,\ Q_v,\ Z_v))$. The message space is $\mathcal{M} \doteq (\{0,1\}^{\kappa})^{v}$.

Register: For each user u_i, the security manager first retrieves the corresponding codeword $\omega^{(i)} \in \mathcal{C}$, and then, for each $j = 1, \ldots, v$, gives u_i one of the two pairs $(\alpha_{j,0}, \beta_{j,0})$ or $(\alpha_{j,1}, \beta_{j,1})$, according to the value of $\omega_j^{(i)}$ (the j-th bit of the codeword $\omega^{(i)}$). The user key of u_i is then set to be $\mathsf{SK}_i \doteq (\alpha_{j,\omega_j^{(i)}},\ \beta_{j,\omega_j^{(i)}})_{j=1,\ldots,v}$. Notice that, for $j = 1, \ldots, v$, $Z_j = c_j P_j = \alpha_{j,\omega_j^{(i)}} P_j + \beta_{j,\omega_j^{(i)}} Q_j$.

Encrypt: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can encrypt a message $m = (m_1 \| \ldots \| m_v) \in \mathcal{M}$ by first selecting random $k_1, \ldots, k_v \in \mathbb{Z}_q$ and then creating a ciphertext $\psi \doteq \langle A_1, B_1, C_1\rangle, \ldots, \langle A_v, B_v, C_v\rangle \in (\mathbb{G}_1^2 \times \{0,1\}^{\kappa})^{v}$ where $A_j \doteq k_j P_j$, $B_j \doteq k_j Q_j$ and $C_j \doteq m_j \oplus H(k_j Z_j)$, $j = 1, \ldots, v$.

Decrypt: Given a ciphertext $\psi = (\langle A_1, B_1, C_1\rangle, \ldots, \langle A_v, B_v, C_v\rangle)$, user u_i computes $k_j Z_j = \alpha_{j,\omega_j^{(i)}} A_j + \beta_{j,\omega_j^{(i)}} B_j)$ and recovers $m_j = C_j \oplus H(k_j Z_j)$, for $j = 1, \ldots, v$.

Trace: To trace a decoder D back to the identity of one of the traitors, the security manager prepares an illegal ciphertext $\hat{\psi} \doteq (\hat{\psi}_1, \ldots, \hat{\psi}_v)$, where each $\hat{\psi}_j$ is constructed as in the tracing algorithm from Section 6.4.1 (*i.e.*, $\hat{\psi}_j \doteq \langle k'_j P_j, k_j Q_j, \hat{m}_j\rangle$, for random $k_j, k'_j \xleftarrow{R} \mathbb{Z}q$ and $\hat{m}_j \xleftarrow{R} \{0,1\}^{\kappa}$). Let $m \doteq (m_1 \| \ldots \| m_v)$ be the plaintext output by D when fed with the ciphertext $\hat{\psi}$.

The security manager forms a "traitor codeword" $\hat{\omega} \doteq (\hat{\omega}^{(1)}, \ldots, \hat{\omega}^{(v)}) \in \{0, 1, \text{‘?’}\}^{v}$, where each $\hat{\omega}^{(j)}$ is derived from m_j as in the tracing algorithm for the two-user scheme (*i.e.*, $\hat{\omega}^{(j)} \doteq \sigma_j$ if $m_j = \hat{m}_j \oplus H(k'_j \alpha_{j,\sigma_j} P_j + k_j \beta_{j,\sigma_j} Q_j)$ (for $\sigma_j = \{0,1\}$), or $\hat{\omega}^{(j)} \doteq$ ‘?’ otherwise).

At this point, the "traitor codeword" $\hat{\omega}$ is run through the tracing algorithm $\mathcal{T}(r_{\mathcal{C}}, \cdot)$ of the collusion-secure code $\mathcal{C}$ (where $r_{\mathcal{C}}$ are the random coins used by the security manager in generating $\mathcal{C}$). Finally, Trace outputs whichever value in $\{1, \ldots, n, \emptyset\}$ returned by $\mathcal{T}(r_{\mathcal{C}}, \hat{\omega})$.

6.5 The CPP Public-Key Traitor Tracing Scheme

6.5.1 The Two-User Sub-Scheme

We now describe the scheme[7] of [27], which is based on the use of bilinear maps. The key difference from the scheme of [57] is the idea of *proxy quantity*: the security manager selects the master secret key roughly as in [57], but now some secret information is removed from the users' secret keys and a derived value (the proxy quantity) is lifted to the public key.

Setup: Given a security parameter 1^κ, the algorithm works as follows:

> **Step 1:** Generate a κ-bit prime q, two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$. Let P be a generator of $\mathbb{G}_1$ and set $g \doteq e(P, P)$.
>
> **Step 2:** Pick random elements $a, c \in \mathbb{Z}_q^*$, and set $Q \doteq aP$, $h \doteq g^c$. The private key of the security manager is set to be the pair $\mathsf{master}_{\mathsf{TT}} \doteq (a, c)$.
>
> **Step 3:** The security manager selects two linearly independent vectors (α_0, β_0) and (α_1, β_1) in $\mathbb{Z}_q^2$ such that $\alpha_\sigma + a\beta_\sigma = c \bmod q$, for $\sigma \in \{0, 1\}$. It chooses a universal hash function $H : \mathbb{G}_1 \to \{0, 1\}^\kappa$, and set the public key of the scheme to be the tuple $\mathsf{params}_{\mathsf{TT}} \doteq (q, \mathbb{G}_1, \mathbb{G}_2, e, H, g, P, Q, h, \alpha_0 P, \beta_0 P, \alpha_1 P, \beta_1 P)$. The message space is $\mathcal{M} \doteq \{0, 1\}^\kappa$.

$\mathsf{Register}$: The secret key of user u_σ is set to be $\mathsf{SK}_\sigma \doteq (\alpha_\sigma)$. Notice that: $cP = \alpha_\sigma P + \beta_\sigma Q$ and hence $e(cP, P) = e(\alpha_\sigma P, P) \cdot e(\beta_\sigma P, Q)$, for $\sigma \in \{0, 1\}$.

$\mathsf{Encrypt}$: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can encrypt a message $m \in \mathcal{M}$ by first selecting a random $k \in \mathbb{Z}_q$ and then creating the ciphertext $\psi \doteq \langle A, B, C \rangle \in \mathbb{G}_1^2 \times \mathcal{M}$ where

$$A \doteq kP, \quad B \doteq k^2 Q, \quad C \doteq m \oplus H(h^{k^2}).$$

[7]In [27], the authors present two schemes with the same parameters. For conciseness, here we only report the second scheme, which was claimed to also support local public traceability.

Decrypt: Given a ciphertext $\psi = \langle A, B, C \rangle$, user u_σ computes $h^{k^2} = e(\alpha_\sigma A, A) \cdot e(\beta_\sigma P, B)$ and recovers $m = C \oplus H(h^{k^2})$.

Trace: To trace a decoder D back to the identity of the traitor, the tracer picks two distinct random values $k, k' \in \mathbb{Z}_q$, along with a random $\hat{m} \in \mathcal{M}$, and feeds D with the "illegal" ciphertext $\hat{\psi} \doteq \langle k'P, k^2 Q, \hat{m} \rangle$. If the output of D is $\hat{m} \oplus H(e(\alpha_\sigma P, P)^{k'^2} \cdot e(\beta_\sigma P, Q)^{k^2})$, then the algorithm returns the identity u_σ as the traitor; otherwise it outputs $\emptyset$.

In [27], the above two-user scheme is proven secure and traceable (for up to 1 traitor) in the sense of Definitions 62 and 63 under two non-standard assumptions for bilinear groups, respectively called DBDH^2-E and DBDH^1-M in [27] (*cf.* Chapter 2, Section 2.3, Definitions 14 and 13, respectively).

6.5.2 The Multi-User Scheme

Similarly to the multi-user of [57] (*c.f.* Section 6.4), it is possible to concatenate multiple instantiations of the two-user scheme from the previous section via collusion-secure codes to obtain a multi-user scheme.

In this case, however, the use of public proxy quantities are sufficient to decrypt and contain less information about the master secret key. This makes it (seemingly) safe to reuse the same parameters P and Q (in the public key) and the same randomness k (in the ciphertext) for all v components of the multi-user scheme. This (seemingly) results in a significant bonus, as it allows for considerably shorter public keys and ciphertexts.

Setup: Given the security parameters 1^κ, 1^t and ε, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let P be a generator of $\mathbb{G}_1$ and set $g \doteq e(P, P)$. Additionally, generate an (ε, t, n, v)-collusion-secure code $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\}$ over $\{0, 1\}$.

Step 2: Pick random elements $a, c_j \in \mathbb{Z}_q^*$ $(j = 1, \ldots, v)$, and set $Q \doteq aP$, $h_j \doteq g^{c_j}$, $j = 1, \ldots, v$. For each $j = 1, \ldots, v$, compute two linearly independent vectors $(\alpha_{j,0}, \beta_{j,0})$, $(\alpha_{j,1}, \beta_{j,1})$ in $\mathbb{Z}q^2$ such that $\alpha_{j,\sigma} + a\beta_{j,\sigma} = c_j \bmod q$, for $\sigma \in \{0, 1\}$. The private key of the security manager is set to be $\mathsf{master}_{\mathsf{TT}} \doteq (a, (\alpha_{j,0}, \beta_{j,0}, \alpha_{j,1}, \beta_{j,1})_{j=1,\ldots,v})$.

Step 3: Choose a universal hash function $H : \mathbb{G}_1 \to \{0, 1\}^\kappa$, and set the public key to: $\mathsf{params}_{\mathsf{TT}} \doteq (q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e,\ H,\ P,\ Q,\ (h_j,\ \alpha_{j,0},\ \beta_{j,0},\ \alpha_{j,1},\ \beta_{j,1})_{j=1,\ldots,v})$. The message space is $\mathcal{M} \doteq (\{0, 1\}^\kappa)^v$.

Register: For each user u_i, the security manager retrieves the corresponding codeword $\omega^{(i)} \in \mathcal{C}$, and sets the user key of u_i to be: $\mathsf{SK}_i \doteq (\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v}$. Notice that, for $j = 1, \ldots, v$, $c_jP = \alpha_{j,\omega_j^{(i)}}P + \beta_{j,\omega_j^{(i)}}Q$ and hence, $h_j = e(c_jP, P) = e(\alpha_{j,\omega_j^{(i)}}P, P) \cdot e(\beta_{j,\omega_j^{(i)}}P, Q)$.

Encrypt: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can encrypt a message $m = (m_1\| \ldots \|m_v) \in \mathcal{M}$ by first selecting a random $k \in \mathbb{Z}_q$ and then creating a ciphertext $\psi \doteq \langle\ A, B, (C_1, \ldots, C_v)\ \rangle \in \mathbb{G}_1^2 \times \mathcal{M}$, where $A \doteq kP$, $B \doteq k^2Q$ and $C_j \doteq m_j \oplus H(h_j^{k^2})$, $j = 1, \ldots, v$.

Decrypt: Given a ciphertext $\psi = \langle A, B, (C_1, \ldots, C_v)\rangle \in \mathbb{G}_1^2 \times \mathcal{M}$, user u_i computes (for $j = 1, \ldots, v$) the mask $h_j^{k^2} = e(\alpha_{j,\omega_j^{(i)}}A, A) \cdot e(\beta_{j,\omega_j^{(i)}}P, B)$ and then recovers each m_j as $m_j = C_j \oplus H(h_j^{k^2})$.

Trace: Although [27] present a tracing algorithm only for their two-user scheme, the authors suggested therein that their multi-user scheme inherits the tracing capabilities of [57]. In particular, we sketch here the obvious necessary modifications to the Trace algorithm in Section 6.4.2: the illegal ciphertext has the form $\hat{\psi} \doteq \langle k'P, k^2Q, (\hat{m}_1, \ldots, \hat{m}_v)\rangle$, where $k, k' \xleftarrow{R} \mathbb{Z}q$, and each $\hat{m}_j$ is random in $\{0, 1\}^\kappa$; and the "traitor codeword" $\hat{\omega} \doteq (\hat{\omega}^{(1)}, \ldots, \hat{\omega}^{(v)})$, is constructed from D's response $m \doteq (m_1\| \ldots \|m_v)$ by defining each $\hat{\omega}^{(j)} \in \{0, 1, \text{‘?’}\}$ as in the tracing for the two-user scheme (*i.e.*, $\hat{\omega}^{(j)} \doteq \sigma_j$ if $m_j = \hat{m}_j \oplus H(e(\alpha_{j,\sigma_j}P, P)^{k'^2} \cdot e(\beta_{j,\sigma_j}P, Q)^{k^2})$ (for $\sigma_j = \{0, 1\}$), or $\hat{\omega}^{(j)} \doteq$ ‘?’ otherwise).

6.6 On the Query Complexity of KY Black-Box Tracing

In Section 6.4.2, we reported the multi-user scheme of [57], which includes a black-box tracing algorithm making a single query to the pirate decoder D. Below we show that such algorithm is broken, and we present a simple pirate strategy that allows a coalition of just $2 < t$ users to escape tracing with probability 1. We also propose a variation of their black-box tracing algorithm, which requires v queries but is successful in tracing up to the desired threshold of traitors, thus suggesting that the query complexity of black-box tracing in [57] is higher than what claimed therein.

6.6.1 A Simple Untraceable Pirate Strategy

Consider the coalition of 2 users, which for simplicity we will suppose associated with the first two codewords $\omega^{(1)}, \omega^{(2)}$ of $\mathcal{C}$. Since $\omega^{(1)} \neq \omega^{(2)}$, they must differ in at least one of their v bits, say the first bit.

This means that by pooling their secret keys, the two traitors can construct a pirate decoder D containing both user-keys $(\alpha_{1,0}, \beta_{1,0}), (\alpha_{1,1}, \beta_{1,1})$ for the two-user sub-scheme associated to index 1, plus at least one user-key for each of the remaining $(v - 1)$ components. When given a ciphertext $\psi \doteq \langle \psi_1, \ldots, \psi_v \rangle$, D starts by decrypting ψ_1 twice: once using $(\alpha_{1,0}, \beta_{1,0})$, and then again using $(\alpha_{1,1}, \beta_{1,1})$. If the two resulting plaintexts coincide, then D decrypts the rest of ψ and output the resulting message; otherwise, D can conclude that it is being traced, and can just output a predetermined message (*e.g.,* the all-zero message).

Notice that D perfectly decrypts ciphertext distributed according to algorithm $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, \cdot)$ since, by correctness of decryption, D's "integrity" check will always pass on a valid ciphertext. Moreover, D escapes tracing with probability 1, since the Trace algorithm of [57] prepares the invalid ciphertext $\hat{\psi}$ by concatenating invalid ciphertexts $\hat{\psi}_j$ for each of the v components of the scheme. This will result in different decryptions of $\hat{\psi}_1$ under $(\alpha_{1,0}, \beta_{1,0})$ and $(\alpha_{1,1}, \beta_{1,1})$, and thus D will reply with a plaintext containing no information about the identities of the traitors.

6.6.2 The Fix

The problem with the Trace algorithm of [57] is that it implicitly assumed that pirate decoders would decrypt each component of the ciphertext independently from each other, which clearly does not need to be the case. Bearing this in mind, the fix is immediate: it suffices for Trace to iteratively query the decoder with v ciphertexts, each constructed to be invalid in just one component, but valid elsewhere. Now, the independence of the v component sub-schemes implies that D will be unable to tell valid and invalid ciphertexts apart, unless it possesses both user-keys for the single sub-scheme "under testing." As a consequence, Trace will end up extracting a traitor codeword from D with at most t unreadable marks '?', and thus the tracing algorithm $\mathcal{T}(\cdot, \cdot)$ of the collusion-secure code $\mathcal{C}$ will successfully recover the identity of one of the traitor (with probability $1 - \varepsilon$).

6.6.3 Consequences for the Multi-User CPP Scheme

Being based on the techniques of [57], the multi-user scheme of [27] inherits the problem pointed out in Section 6.6.1. As it turns out, however, in this case the consequences are more severe. In particular, the easy fix that we proposed for the scheme of [57] in Section 6.6.2 does not apply: interestingly, the higher correlation between the parameters used in the v components of the scheme of [27], which proved crucial to attain optimal transmission rate, at the same time poses a serious impediment to black-box tracing.

Indeed, ciphertexts in the multi-user scheme of [27] (*cf.* Section 6.5.2) have the form $\psi \doteq \langle kP,\ k^2Q,\ (C_1,\ \ldots,\ C_v)\rangle$, in which the same "randomization" values kP, k^2Q are used for all the v two-user sub-schemes. Hence, it is not possible to make the ciphertext invalid in just one component, while preserving its validity in the remaining $(v - 1)$ ones (which was the idea behind our fix in Section 6.6.2). Therefore, it seems that the scheme of [27], as given, does not support black-box tracing. Since the notion of local public traceability is only meaningful in the black-box setting, this also voids the claimed traceability features of the multi-user scheme of [27].

To salvage black-box tracing and local public traceability, one could modify the scheme of [27] and revert to the "parallel" composition of sub-schemes (exactly as in [57]), thus "undoing" the optimization that enabled short ciphertexts. The resulting scheme, however, would just be a variant of [57] with the same parameters, but with the additional need of bilinear maps and reliance on non-standard bilinear-related assumptions.

As a result, it seems appropriate to regard the multi-user scheme of [27] as a scheme with optimal transmission rate, but with only non-black-box tracing and no public traceability features.

6.7 Black-Box Traitor Tracing with Optimal Transmission Rate

6.7.1 The Two-User Sub-Scheme

Setup: Given a security parameter 1^κ, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Choose an arbitrary generator $P \in \mathbb{G}_1$.

Step 2: Pick random elements $a, b, c \in \mathbb{Z}q^*$, and set $Q \doteq aP, R \doteq bP, h \doteq e(P, cP)$. The private key of the security manager is set to be $\mathsf{master}_{\mathsf{TT}} \doteq (a, b, c)$.

Step 3: The security manager selects two linearly independent vectors (α_0, β_0) and (α_1, β_1) in $\mathbb{Z}_q$ such that $b\alpha_\sigma + a\beta_\sigma = c \bmod q$, for $\sigma \in \{0, 1\}$. Finally, it chooses a universal hash function $H : \mathbb{G}_1 \to \{0, 1\}^\kappa$ and set the public key of the scheme to be the tuple:

$$\mathsf{params}_{\mathsf{TT}} \doteq (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P).^{8}$$

The associated message space is $\mathcal{M} \doteq \{0,1\}^{\kappa}$.

Register: The secret key of user u_σ is set to be $\mathsf{SK}_\sigma \doteq \alpha_\sigma$. Notice that, $cP = \alpha_\sigma R + \beta_\sigma Q$ and hence $h = e(P, cP) = e(P, R)^{\alpha_\sigma} \cdot e(\beta_\sigma P, Q)$, for $\sigma \in \{0,1\}$.

Encrypt: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can encrypt a message $m \in \mathcal{M}$ by first selecting a random $k \in \mathbb{Z}_q$ and then creating the ciphertext $\psi \doteq \langle A, B, C \rangle \in \mathbb{G}_2 \times \mathbb{G}_1 \times \mathcal{M}$ where

$$A \doteq e(P,R)^k, \quad B \doteq kQ, \quad C \doteq m \oplus H(h^k).$$

Decrypt: Given a ciphertext $\psi = \langle A, B, C \rangle$, user u_σ computes $h^k = A^{\alpha_\sigma} \cdot e(\beta_\sigma P, B)$ and recovers $m = C \oplus H(h^k)$. Correctness of the decryption algorithm is clear by inspection.

Trace: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can trace a decoder D back to the identity of the traitor, as follows: First, pick two distinct random values $k, k' \in \mathbb{Z}_q$, along with a random $\hat{m} \in \mathcal{M}$; then, feed D with the "illegal" ciphertext $\hat{\psi} \doteq \langle e(P,R)^{k'}, kQ, \hat{m} \rangle$. If the output of D is $\hat{m} \oplus H(e(P, \alpha_\sigma R)^{k'} \cdot e(Q, \beta_\sigma P)^k)$, then return u_σ as the traitor's identity; otherwise, output $\emptyset$.

Before moving on to proving the security and traceability of our two-user scheme in the sense of Definitions 62 and 63 (*cf. Section 6.3*), we remark that Trace does not require knowledge of the master secret key $\mathsf{master}_{\mathsf{TT}}$, and thus, like the two-user scheme of [27], it supports *Public Traceability* (*cf.* Definition 61). Also, notice that decryption requires only one pairing computation.

[8]Note that there is no need to explicitly include h in the public key, as it can be derived as $h = e(P, \alpha_\sigma R) \cdot e(Q, \beta_\sigma P)$. Caching the value of h, however, is recommendable when public storage is not at a premium, as that would save two pairing computations during encryption.

6.7.2 Indistinguishability under Chosen-Plaintext Attack

Theorem 64. *Under the* DBDH *assumption for* $(\mathbb{G}_1, \mathbb{G}_2)$*, the scheme in Section 6.7.1 is secure w.r.t. indistinguishability under chosen-plaintext attack (*c.f. *Definition 11 and Definition 62).*

Proof. Let us assume that the scheme does not satisfy Definition 62, *i.e*, there is an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that, given the public key

$$\mathsf{params}_{\mathsf{TT}} = (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P)$$

can break the scheme with advantage ε_{ind}. We then construct an algorithm $\mathcal{B}$ (whose running time is polynomially related to $\mathcal{A}$'s) that breaks the DBDH assumption with probability $\varepsilon_{\mathsf{DBDH}} = \varepsilon_{\text{ind}}$.

Algorithm $\mathcal{B}$ is given as input an instance (P', xP', yP', zP', h') of the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$; its task is to determine whether $h' = e(P', P')^{xyz}$ or h' is a random element in $\mathbb{G}_2$. $\mathcal{B}$ proceeds as follows:

Setup: $\mathcal{B}$ lets $P \doteq xP'$ and $Q \doteq P'$. Then, $\mathcal{B}$ picks $r \overset{R}{\leftarrow} \mathbb{Z}q^*$, and sets $R \doteq rQ$. $\mathcal{B}$ now chooses $\beta_0, \beta_1 \overset{R}{\leftarrow} \mathbb{Z}q^*$ and computes $\beta_0 P$ and $\beta_1 P$. Then, $\mathcal{B}$ lets $\alpha_0 R \doteq zP'$ and $\alpha_1 R \doteq \alpha_0 R + \beta_0 Q - \beta_1 Q$. $\mathcal{B}$ can now set $\mathsf{params}_{\mathsf{TT}} \doteq (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P)$ and send it to $\mathcal{A}_1$.

Challenge: $\mathcal{A}_1$ outputs two message m_0, m_1 on which it wishes to be challenged, along with some state τ to be passed to $\mathcal{A}_2$. To prepare the ciphertext, $\mathcal{B}$ lets

$$kQ \doteq yP', \qquad e(P, R)^k \doteq e(P, kQ)^r, \qquad h^k \doteq h' \cdot e(\beta_0 P, kQ).$$

(Notice that this implicitly defines $k = y$.) Then, $\mathcal{B}$ picks random $b^* \in \{0, 1\}$, and sends the ciphertext $(e(P, R)^k,\ kQ,\ m_{b^*} \oplus H(h^k))$ as challenge to $\mathcal{A}_2$ (along with the state τ).

Guess: Algorithm $\mathcal{A}_2$ outputs a guess $b' \in \{0, 1\}$. $\mathcal{B}$ returns 1 if $b' = b^*$ and 0 otherwise.

If $h' = e(P', P')^{xyz}$, $\mathcal{A}_2$ gets a valid encryption of m_{b^*}, since then:

$$h^k = h' \cdot e(\beta_0 P, kQ) = e(P', P')^{xyz} \cdot e(\beta_0 P, Q)^k = e(xP', zP')^y \cdot e(\beta_0 P, Q)^k =$$
$$= e(P, \alpha_0 R)^y \cdot e(\beta_0 P, Q)^k = e(P, \alpha_0 R)^k \cdot e(\beta_0 P, Q)^k = [e(P, R)^{\alpha_0} \cdot e(\beta_0 P, Q)]^k.$$

So when $\mathcal{B}$'s input comes from the DBDH distribution, $b' = b^*$ holds with probability $\varepsilon_{\text{ind}} + 1/2$.

Otherwise, since H is randomly chosen from a universal hash function family, the challenge is independent from m_{b^*}. In fact, when h' is randomly chosen, the message m_{b^*} is encrypted with a totally random value, independent from b^*. Thus, in this case $b' = b^*$ holds with probability $1/2$.

It follows that adversary $\mathcal{B}$ breaks the DBDH assumption with advantage $\varepsilon_{\mathsf{DBDH}} = \varepsilon_{\text{ind}}$. $\square$

6.7.3 Traceability

To prove the security of the Trace algorithm, we first observe that security of the encryption functionality (*cf.* Theorem 64) implies that any efficient pirate decoder D that successfully inverts the $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, \cdot)$ functionality must "incorporate" knowledge of at least one user key. Indeed, if D did not somehow encode knowledge of any user key SK_σ, yet it successfully inverted $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, \cdot)$ (almost) everywhere (in probabilistic polynomial-time), then we would have an efficient algorithm distinguishing between ciphertexts of any two distinct messages based only on the system's public key $\mathsf{params}_{\mathsf{TT}}$, contradicting Theorem 64.

Then, the goal of the Trace algorithm can be recast as that of identifying which secret information is embedded within D that enables it to go beyond what's (efficiently) doable based only on $\mathsf{params}_{\mathsf{TT}}$. In principle, any $(\alpha, \Pi) \in \mathbb{Z}q \times \mathbb{G}_1$ such that $h = e(P, R)^\alpha \cdot e(\Pi, Q)$ would suffice, as D could then recover the mask h^k from $\psi \doteq \langle e(P, R)^k, kQ, m \oplus H(h^k)\rangle$ as $h^k = (e(P, R)^k)^\alpha \cdot e(\Pi, kQ)$. However, below we prove that, under the CDH assumption (*cf.* Chapter 2, Section 2.3), the only such pair (α,Π) that the traitor u_σ can obtain from the public key and his/her secret key is $(\alpha_\sigma, \beta_\sigma P)$:

Lemma 65. *Under the* CDH *assumption in* $\mathbb{G}_1$*, given the public key* $\mathsf{params_{TT}} = (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P)$ *and the secret key* $\mathsf{SK}_\sigma \doteq \alpha_\sigma$ *of user* u_σ*, it is computational infeasible to construct a pair* $(\alpha, \Pi) \in \mathbb{Z}q \times \mathbb{G}_1$ *such that* $h = e(P, R)^\alpha \cdot e(\Pi, Q)$*, but* $\alpha \neq \alpha_\sigma$*.*

Proof. Assume for simplicity that $\sigma = 0$. We proceed by contradiction, *i.e.* let us assume that there exists an adversary $\mathcal{A}$ that, given the public key $\mathsf{params_{TT}}$ and the secret key α_0, is able to compute $(\alpha, \Pi) \in \mathbb{Z}q \times \mathbb{G}_1$ such that $h = e(P, R)^\alpha \cdot e(\Pi, Q)$ but $\alpha \neq \alpha_0$. We then construct an algorithm $\mathcal{B}$ (whose running time is polynomially related to $\mathcal{A}$'s) that solves the CDH problem in $\mathbb{G}_1$.

On input a random instance (P', xP', yP') of the CDH problem in $\mathbb{G}_1$, $\mathcal{B}$ proceeds as follows:

Setup: $\mathcal{B}$ lets $P \doteq xP'$, $Q \doteq P'$ and $R \doteq yP'$. $\mathcal{B}$ then chooses $\alpha_0, \beta_0, \beta_1 \overset{R}{\leftarrow} \mathbb{Z}q^*$ and computes $\alpha_0 R$, $\beta_0 P$ and $\beta_1 P$. Then, $\mathcal{B}$ lets $\alpha_1 R \doteq \alpha_0 R + \beta_0 Q - \beta_1 Q$. This implicitly defines $h = e(P, R)^{\alpha_0} \cdot e(\beta_0 P, Q) = e(P, R)^{\alpha_1} \cdot e(\beta_1 P, Q)$. $\mathcal{B}$ can now set $\mathsf{params_{TT}} \doteq (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P)$ and send it to $\mathcal{A}_1$, along with the secret key $\mathsf{SK}_0 = \alpha_0$ of user u_0.

Attack: $\mathcal{A}$ outputs a pair (α, Π) such that $h = e(P, R)^\alpha \cdot e(\Pi, Q)$, but $\alpha \neq \alpha_0$.

Break: $\mathcal{B}$ outputs $(\alpha_0 - \alpha)^{-1} \cdot (\Pi - \beta_0 x P')$.

If $\mathcal{A}$'s output is correct, then $\mathcal{B}$'s output is xyP'. Indeed, writing $\Pi = \beta P$, for some unknown $\beta \in \mathbb{Z}q^*$, the fact that $e(P, R)^\alpha \cdot e(\beta P, Q) = h = e(P, R)^{\alpha_0} \cdot e(\beta_0 P, Q)$, implies that $e(P, \alpha R + \beta Q) = e(P, \alpha_0 R + \beta_0 Q)$, and from the injectivity of $e(P, \cdot)$, we get:

$$\alpha_0 R + \beta_0 Q = \alpha R + \beta Q \Leftrightarrow \alpha_0 y P' + \beta_0 P' = \alpha y P' + \beta P' \Leftrightarrow$$
$$\Leftrightarrow \alpha_0 y + \beta_0 = \alpha y + \beta \Leftrightarrow y = (\beta - \beta_0) \cdot (\alpha_0 - \alpha)^{-1}.$$

Hence, the target CDH value can be written as:

$$xyP' = ((\beta - \beta_0)(\alpha_0 - \alpha)^{-1}) xP' = (\alpha_0 - \alpha)^{-1}(\beta x P' - \beta_0 x P') =$$
$$= (\alpha_0 - \alpha)^{-1}(\Pi - \beta_0 x P').$$

□

An immediate corollary of the above lemma is a sort of "non-incrimination" property: a traitor u_σ cannot frame the innocent user $u_{1-\sigma}$ by constructing a pirate decoder containing $\mathsf{SK}_{1-\sigma}$:

Corollary 66 (Non-Incrimination)**.** *Under the* CDH *assumption, given the public key* $\mathsf{params}_{\mathsf{TT}} = (q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e,\ H,\ P,\ Q,\ R,\ \alpha_0 R, \beta_0 P,\ \alpha_1 R, \beta_1 P)$ *and the secret key* $\mathsf{SK}_\sigma = \alpha_\sigma$ *of user* u_σ*, it is computationally infeasible to construct the secret key* $\mathsf{SK}_{1-\sigma} \doteq \alpha_{1-\sigma}$ *of user* $u_{1-\sigma}$*.*

Another consequence of Lemma 65 is that a pirate decoder D created by u_σ decrypts a ciphertext $\psi \doteq \langle A, B, C\rangle$ as $m \doteq C \oplus H(A^{\alpha_\sigma} \cdot e(\beta_\sigma P, B))$. However, this holds (except with negligible probability) when ψ is properly created, whereas the Trace algorithm from Section 6.7.1 queries D on "probe" ciphertexts which are not distributed according to $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{TT}}, \cdot)$:

Definition 67 (Valid and Probe Ciphertexts)**.** *A ciphertext* $\psi = \langle A, B, C\rangle$ *is*

- ***valid****, if* $A = e(P,R)^k, B = kQ$*, for random* $k \in \mathbb{Z}_q$*;*
- ***probe****, if* $A = e(P,R)^{k'}, B = kQ$*, for random* $k, k' \in \mathbb{Z}_q$*.*

Nevertheless, in Lemma 68 we prove that D cannot distinguish between the two cases, and thus D replies to a probe with the "plaintext" $C \oplus H((e(P,R)^{k'})^{\alpha_\sigma} \cdot e(\beta_\sigma P, kQ))$. It follows that the Trace algorithm from Section 6.7.1 correctly identifies the traitor u_σ, except with negligible probability, as required by Definition 63.

Lemma 68 (Indistinguishability of Valid *vs.* Probe Ciphertexts)**.** *Under the* DBDH *assumption for* $(\mathbb{G}_1, \mathbb{G}_2)$*, given the public key*

$$\mathsf{params}_{\mathsf{TT}} = (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P)$$

and the secret key $\mathsf{SK}_\sigma = \alpha_\sigma$ *of user* u_σ*, it is infeasible to distinguish a valid ciphertext from a probe.*

Proof. For simplicity, assume $\sigma = 0$. We proceed by contradiction: assume there is an adversary $\mathcal{A}$ that, given the public key

$$\mathsf{params}_{\mathsf{TT}} = (q, \mathbb{G}_1, \mathbb{G}_2, e, H, P, Q, R, \alpha_0 R, \beta_0 P, \alpha_1 R, \beta_1 P)$$

and the secret key α_0 of user u_0, can distinguish valid ciphertexts from probes with probability ε. We then construct an algorithm $\mathcal{B}$ (whose running time is polynomially related to $\mathcal{A}$'s) that breaks the DBDH assumption with probability $\varepsilon_{\mathsf{DBDH}} = \varepsilon$.

Algorithm $\mathcal{B}$ is given as input an instance (P', xP', yP', zP', h') of the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$; its task is to determine whether $h' = e(P', P')^{xyz}$ or h' is a random element in $\mathbb{G}_2$. $\mathcal{B}$ proceeds as follows:

Setup: $\mathcal{B}$ lets $P \doteq xP'$, $Q \doteq P'$, $R \doteq yP'$, chooses $\alpha_0, \beta_0, \beta_1 \overset{R}{\leftarrow} \mathbb{Z}q^*$ and computes $\alpha_0 R$, $\beta_0 P$ and $\beta_1 P$. $\mathcal{B}$ also lets $\alpha_1 R \doteq \alpha_0 R + \beta_0 Q - \beta_1 Q$, and $\mathsf{params}_{\mathsf{TT}} \doteq (q$, $\mathbb{G}_1$, $\mathbb{G}_2$, e, H, P, Q, R, $\alpha_0 R$, $\beta_0 P$, $\alpha_1 R$, $\beta_1 P)$. Then, $\mathcal{B}$ lets $kQ \doteq zP'$ (which implicitly defines $k = z$) and prepares a challenge ciphertext $\hat{\psi}$ by picking $\hat{m} \in \mathcal{M}$ and setting $\hat{\psi} \doteq \langle h', kQ, \hat{m} \rangle$. At this point, $\mathcal{B}$ feeds $\mathcal{A}$ with $\mathsf{params}_{\mathsf{TT}}$, $\hat{\psi}$, and α_0.

Attack: $\mathcal{A}$ returns her guess to whether $\hat{\psi}$ is a valid ciphertext or a probe (w.r.t. the public key $\mathsf{params}_{\mathsf{TT}}$).

Break: $\mathcal{B}$ outputs yes or no accordingly.

If $h' = e(P', P')^{xyz}$, then $\mathcal{A}$ gets a valid ciphertext since $h' = e(xP', yP')^z = e(P, R)^k$. Otherwise, h' is a random value, totally independent from k, and thus $\hat{\psi}$ is a probe. Therefore, $\mathcal{B}$ breaks the DBDH assumption with the same advantage as $\mathcal{A}$'s *i.e.,* $\varepsilon_{\mathsf{DBDH}} = \varepsilon$. □

6.7.4 The Multi-User Scheme

To generalize the two-user scheme of Section 6.7.1, we start from the following considerations. During tracing, we make v queries to the decoder; in each query, one component of the ciphertext ought to be invalid, whereas the remaining $(v - 1)$ components should be valid. In this way, if the decoder does not have both keys for the invalid component, it will be unable to tell that the ciphertext is invalid.

A straightforward way to achieve this independence of the components is to have v independent copies of the two-user scheme in Section 6.7.1, and use independent randomness in the encryption. This, however, would yield public-storage rate 4 and transmission rate 3, which is no better than what already attained by [57].

To get ciphertext rate 1, the idea is that each ciphertext will include a "special" component $\ell \in \{1, \ldots, v\}$ which will be encrypted using an instance of the two-user scheme of Section 6.7.1 specific to the ℓ-th component; the remaining $(v-1)$ components, on the other hand will be encrypted using a "shared" two-user scheme, as in the scheme of [27] (*cf.* also Section 6.5.2).

Setup: Given the security parameters 1^κ, 1^t and ε, the algorithm works as follows:

Step 1: Generate a κ-bit prime q, two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Generate an (ε, t, n, v)-collusion-secure code $\mathcal{C} = \{\omega^{(1)}, \ldots, \omega^{(n)}\}$.

Step 2a: Generate v independent copies of the 2-user scheme described in Section 6.7.1 (call these copies the *special* schemes). In particular, for $j = 1, \ldots, v$, let P_j be a generator of $\mathbb{G}_1$; pick random elements $a_j, b_j, c_j \in \mathbb{Z}_q^*$, and set $Q_j \doteq a_j P_j$, $R_j \doteq b_j P_j$, $h_j \doteq e(P_j, c_j P_j)$. Also, for $j = 1, \ldots, v$, compute linearly independent vectors $(\alpha_{j,0}, \beta_{j,0})$, $(\alpha_{j,1}, \beta_{j,1}) \in \mathbb{Z}q^2$ such that $b_j \alpha_{j,\sigma} + a_j \beta_{j,\sigma} = c_j \bmod q$, for $\sigma \in \{0, 1\}$.

Step 2b: Generate one more independent copy of the 2-user scheme of Section 6.7.1, in which we additionally select v values for h (call this the *shared* scheme). At a high level, the shared scheme can be thought of as v parallel copies of the 2-user scheme of Section 6.7.1, sharing the same values P, Q and R. More precisely, draw $P \overset{R}{\leftarrow} \mathbb{G}_1$, $a, b \overset{R}{\leftarrow} \mathbb{Z}q^*$, and set $Q \doteq aP$, and $R \doteq bP$; then, for each $j = 1, \ldots, v$, select $\bar{c}_j \in \mathbb{Z}_q^*$ and set $\bar{h}_j \doteq e(P, \bar{c}_j P)$. Also, for each $j = 1, \ldots, v$, compute two linearly independent vectors $(\bar{\alpha}_{j,0}, \bar{\beta}_{j,0})$, $(\bar{\alpha}_{j,1}, \bar{\beta}_{j,1})$ in $\mathbb{Z}q^2$ such that $b\bar{\alpha}_{j,\sigma} + a\bar{\beta}_{j,\sigma} = c_j \bmod q$, for $\sigma \in \{0,1\}$.

Step 2c: The master secret key of the security manager is set to be:

$$\mathsf{master}_{\mathsf{TT}} \doteq ((a_j, b_j, (\alpha_{j,0}, \beta_{j,0}, \alpha_{j,1}, \beta_{j,1}))_{j=1,\ldots,v}, a, b, (\bar{\alpha}_{j,0}, \bar{\beta}_{j,0}, \bar{\alpha}_{j,1}, \bar{\beta}_{j,1})_{j=1,\ldots,v}).$$

Step 3: Choose a universal hash function $H : \mathbb{G}_1 \to \{0,1\}^\kappa$, and set $\mathsf{params}_{\mathsf{TT}} \doteq (H, (P_j, Q_j, R_j,\ \alpha_{j,0}R_j,\ \beta_{j,0}P_j,\ \alpha_{j,1}R_j,\ \beta_{j,1}P_j)_{j=1,\ldots,v},\ P, Q, R,\ (\bar{\alpha}_{j,0}R,\ \bar{\beta}_{j,0}P,\ \bar{\alpha}_{j,1}R,\ \bar{\beta}_{j,1}P)_{j=1,\ldots,v})$. The associated message space is $\mathcal{M} \doteq (\{0,1\}^\kappa)^v$.

Register: For each user u_i, the security manager first retrieves the corresponding codeword $\omega_i \in \mathcal{C}$ and sets the secret key of user u_i to:

$$\mathsf{SK}_i \doteq ((\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v}, (\bar{\alpha}_{j,\omega_j^{(i)}})_{j=1,\ldots,v}).$$

Notice that, for $j = 1, \ldots, v$, it holds that:

$$c_jP_j = \alpha_{j,\omega_j^{(i)}}R_j + \beta_{j,\omega_j^{(i)}}Q_j$$
$$\bar{c}_jP = \bar{\alpha}_{j,\omega_j^{(i)}}R + \bar{\beta}_{j,\omega_j^{(i)}}Q$$

and hence

$$h_j = e(P_j, c_jP_j) = e(P_j, \alpha_{j,\omega_j^{(i)}}R_j) \cdot e(\beta_{j,\omega_j^{(i)}}P_j, Q_j)$$
$$\bar{h}_j = e(P, \bar{c}_jP) = e(P, \bar{\alpha}_{j,\omega_j^{(i)}}R) \cdot e(\bar{\beta}_{j,\omega_j^{(i)}}P, Q).$$

Encrypt: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can encrypt a message $m = (m_1\| \ldots \|m_v) \in \mathcal{M}$ as follows: First, select $\ell \overset{R}{\leftarrow} \{1, \ldots, v\}$ and $k_\ell \overset{R}{\leftarrow} \mathbb{Z}_q$, and compute the special component of the ciphertext $(A_\ell, B_\ell, C_\ell) \in \mathbb{G}_2 \times \mathbb{G}_1 \times \{0,1\}^\kappa$, where $A_\ell \doteq e(P_\ell, R_\ell)^{k_\ell}$, $B \doteq k_\ell Q_\ell$ and $C_\ell \doteq m_\ell \oplus H(h_\ell^{k_\ell})$. Then, select $k \overset{R}{\leftarrow} \mathbb{Z}_q$, and compute the remaining pieces of the ciphertext as: $(A, B, C_1, \ldots, C_{\ell-1}, C_{\ell+1}, \ldots, C_v)$, where $A \doteq e(P,R)^k$, $B \doteq kQ$, and $C_j \doteq m_j \oplus H(\bar{h}_j^k)$, for $j = 1, \ldots, v$, $j \neq \ell$. The ciphertext is set to be the tuple $\psi \doteq \langle \ell, A_\ell, B_\ell, A, B, C_1, \ldots, C_v \rangle$

Decrypt: Given a ciphertext $\psi = \langle \ell, A_\ell, B_\ell, A, B, C_1, \ldots, C_v \rangle \in \mathbb{Z} \times (\mathbb{G}_2 \times \mathbb{G}_1)^2 \times \mathcal{M}$, u_i computes:

$$h_\ell^{k_\ell} = (A_\ell)^{\alpha_{j,\omega_j^{(i)}}} \cdot e(\beta_{j,\omega_j^{(i)}} P_j, B_\ell)$$
$$\bar{h}_j^{k} = (A)^{\bar{\alpha}_{j,\omega_j^{(i)}}} \cdot e(\bar{\beta}_{j,\omega_j^{(i)}} P, B) \quad (j = 1, \ldots, v, j \neq \ell)$$

recovers $m_\ell = C_\ell \oplus H(h_\ell^{k_\ell})$ and $m_j = C_j \oplus H(\bar{h}_j^k)$ (for $j \in \{1, \ldots, v\} \setminus \{\ell\}$) and outputs $m \doteq (m_1 \| \ldots \| m_v)$.

Trace: Given $\mathsf{params}_{\mathsf{TT}}$, anybody can extract the "traitor codeword" $\hat{\omega} \doteq (\hat{\omega}^{(1)}, \ldots, \hat{\omega}^{(v)}) \in \{0, 1, \text{'?'}\}^v$ from a decoder D by making v queries to D. At a high level, the idea is to iteratively derive each $\hat{\omega}^{(j)}$ by feeding D with an invalid ciphertext that looks valid in the "shared" components, but is actually a *probe* on the j-th "special" component. In this way, if D contains only one of the two user-keys for the j-th "special" two-user component (say, α_{j,σ_j}), it will reply with a plaintext which reveals the value of σ_j. More in detail, the probe ciphertext to extract $\hat{\omega}^{(j)}$ from D is: $\hat{\psi}^{(j)} \doteq \langle j, \hat{A}_j, \hat{B}_j, A^{(j)}, B^{(j)}, C_1^{(j)}, \ldots, C_v^{(j)} \rangle$, where $k_j, k'_j, k^{(j)}$ are drawn from $\mathbb{Z}q$, $C_1^{(j)}, \ldots, C_v^{(j)}$ are drawn from $\{0,1\}^\kappa$, and

$$\hat{A}_j \doteq e(P_j, R_j)^{k'_j}, \qquad \hat{B}_j \doteq k_j Q_j, \qquad A^{(j)} \doteq e(P, R)^{k^{(j)}}, \qquad B^{(j)} \doteq k^{(j)} Q.$$

Let $m \doteq (m_1 \| \ldots \| m_v)$ be the plaintext output by D when fed with the ciphertext $\hat{\psi}^{(j)}$. Then $\hat{\omega}^{(j)}$ is derived from m_j as in the tracing algorithm for the two-user scheme *i.e.*,

$$\hat{\omega}^{(j)} \doteq \begin{cases} \sigma_j & \text{if } m_j = C_j^{(j)} \oplus H(e(\alpha_{j,\sigma_j} R_j, P_j)^{k'_j} \cdot e(\beta_{j,\sigma_j} P_j, Q_j)^{k_j}), \\ \text{'?'} & \text{otherwise.} \end{cases}$$

At this point, the "traitor codeword" $\hat{\omega}$ is handed to the tracer, who (knowing the random coins $r_{\mathcal{C}}$ used in generating $\mathcal{C}$) can run it through the tracing algorithm $\mathcal{T}(r_{\mathcal{C}}, \cdot)$ of the collusion-secure code $\mathcal{C}$, thus obtaining a value in $\{1, \ldots, n, \emptyset\}$, which is the output of Trace.

Before moving on to proving the security and traceability of our multi-user scheme, we make few remarks.

First, following [66] (and [57, 27]), we assume that it is sensible to focus the tracing effort only on pirate decoders that work with probability very close to 1, since in concrete applications (*e.g.,* distribution of multimedia content) pirate decoders with sub-optimal decryption capabilities are not likely to attract a significant share of the content distribution network's customer base. However, as already noted in [57], any constant-rate traitor tracing scheme can be made effective against decoders that decrypt any non-negligible fraction of ciphertexts by simply pre-processing plaintexts with an all-or-nothing transform (AONT) [70, 25] (*cf.* [57] for details).

Second, since the Trace algorithm needs $\mathsf{master}_{\mathsf{TT}}$ only in the off-line phase, which does not access the pirate decoder and is much less computation-intensive,[9] our multi-user scheme supports *Local Public Traceability* (*cf.* Definition 61).

Finally, notice that the size of the message blocks can be shrunk to any $\kappa' \leq \kappa$, by choosing a universal hash function $H : \mathbb{G}_1 \to \{0,1\}^{\kappa'}$. This is possible as long as $\kappa' > \log v + \log(1/\varepsilon) = O(\log t + \log\log(n/\varepsilon) + \log(1/\varepsilon))$, which ensures that, during tracing, the probability of a hash collision in any of the v components of the scheme is bounded by ε. For the choice of parameters given in Section 5.1 ($n = 2^{30}$, $\varepsilon = 2^{-30}$, $t = 30$), κ' can be chosen as low as 64 bits.

6.7.5 Indistinguishability under Chosen-Plaintext Attack

Theorem 69. *Under the* DBDH *assumption for* $(\mathbb{G}_1, \mathbb{G}_2)$*, the scheme in Section 6.7.4 is secure w.r.t. indistinguishability under chosen-plaintext attack (*cf. *Chapter 2 Section 2.3 and Definition 62).*

In light of Theorem 64, Theorem 69 reduces to Lemma 70, whose proof follows from a standard hybrid argument:

[9]For the scheme of [79], for example, such computation consists just of a matrix-vector multiplication.

Lemma 70. *If the two-user scheme in Section 6.7.1 is secure w.r.t. indistinguishability under chosen-plaintext attack (*cf. *Theorem 64), then the multi-user scheme in Section 6.7.4 is secure w.r.t. the same notion.*

6.7.6 Traceability

Throughout this section, T denotes the set of indices of the (up to t) traitors. The following definitions aim at capturing the possible secret information that the traitors in T could derive from pooling their secret keys together.

Definition 71 (T-Associated Secret Tuple)**.** *A tuple* $\hat{\Omega} \doteq (\hat{\alpha}_j, \hat{\Pi}_j, \hat{\bar{\alpha}}_j, \hat{\bar{\Pi}}_j)_{j=1,\ldots,v} \in ((\mathbb{G}_2 \times \mathbb{G}_1)^2)^v$ *is* T-associated *to* $\hat{\omega} \in \{0, 1, \text{‘?’}\}^v$ *if for all* $j \in [1, v]$*, either* $\hat{\omega}_j = \text{‘?’}$*, or* $(\hat{\alpha}_j, \hat{\Pi}_j, \hat{\bar{\alpha}}_j, \hat{\bar{\Pi}}) = (\alpha_{j,\hat{\omega}_j}, \beta_{j,\hat{\omega}_j} P_j, \bar{\alpha}_{j,\hat{\omega}_j}, \bar{\beta}_{j,\hat{\omega}_j} P)$*.*

Definition 72 (T-Feasible Secret Tuple)**.** *Let* F_T *be the set of feasible codewords for* T *(*cf. *Definition 6). Define the set of* T*-feasible secret tuple as* $\mathcal{C}(F_T) \doteq \{\hat{\Omega} \in ((\mathbb{G}_2 \times \mathbb{G}_1)^2)^v \mid (\exists \hat{\omega} \in F_T).[\hat{\Omega}$ *is* T*-associated to* $\hat{\omega}]\}$*.*

Lemma 73. *Given the public key* $\mathsf{params}_{\mathsf{TT}} = (q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e,\ H,\ (P_j, Q_j, R_j,\ \alpha_{j,0} R_j,\ \beta_{j,0} P_j,\ \alpha_{j,1} R_j,\ \beta_{j,1} P_j)_{j=1,\ldots,v},\ P, Q, R,\ (\bar{\alpha}_{j,0} R,\ \bar{\beta}_{j,0} P,\ \bar{\alpha}_{j,1} R,\ \bar{\beta}_{j,1} P)_{j=1,\ldots,v})$ *and the secret keys* $\mathsf{SK}_i \doteq ((\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v},\ (\bar{\alpha}_{j,\omega_j^{(i)}})_{j=1,\ldots,v})$*, for* $i \in T$*, constructing a secret tuple* $\hat{\Omega} \doteq (\hat{\alpha}_j, \hat{\Pi}_j, \hat{\bar{\alpha}}_j, \hat{\bar{\Pi}}_j)_{j=1,\ldots,v} \in ((\mathbb{G}_2 \times \mathbb{G}_1)^2)^v$ *such that* $h_j = e(P_j, \hat{\alpha}_j R_j) \cdot e(\hat{\Pi}_j, Q_j)$ *and* $\bar{h}_j = e(P_j, \hat{\bar{\alpha}}_j R_j) \cdot e(\hat{\bar{\Pi}}_j, Q_j)$ *(*$j = 1, \ldots, v$*) but* $\hat{\Omega} \notin \mathcal{C}(F_T)$*, requires breaking the* CDH *problem in* $\mathbb{G}_1$ *or the collusion-secure code* $\mathcal{C}$*.*

Sketch. The proof follows from Lemma 65 and Definitions 6 and 72. □

Corollary 74 (Non-Incrimination)**.** *Under the* CDH *assumption, and assuming collusion-security of the code* $\mathcal{C}$*, given the public key* $\mathsf{params}_{\mathsf{TT}} = (q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e,\ H,\ (P_j, Q_j, R_j,\ \alpha_{j,0} R_j,\ \beta_{j,0} P_j,\ \alpha_{j,1} R_j,\ \beta_{j,1} P_j)_{j=1,\ldots,v},\ P, Q, R,\ (\bar{\alpha}_{j,0} R,\ \bar{\beta}_{j,0} P,\ \bar{\alpha}_{j,1} R,\ \bar{\beta}_{j,1} P)_{j=1,\ldots,v})$ *and the secret keys* $\mathsf{SK}_i \doteq ((\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v},\ (\bar{\alpha}_{j,\omega_j^{(i)}})_{j=1,\ldots,v})$ *(for* $i \in T$*), it is computationally infeasible to construct the secret key* $\mathsf{SK}_{i'}$ *of user* $u_{i'}$*, with* $i' \notin T$*.*

Definition 75 (Valid and ℓ-Probe Ciphertexts)**.** *A ciphertext* $\psi = \langle\ \ell,\ A_\ell,\ B_\ell,\ A,\ B,\ C_1,\ \ldots,\ C_v\rangle$ *is*

- ***valid***, *if* $A_\ell = e(P_\ell, R_\ell)^{k_\ell}, B_\ell = k_\ell Q_\ell$, $A = e(P, R)^k, B = kQ$, *for random* $k, k_\ell \in \mathbb{Z}_q$*;*
- ***ℓ-probe***, *if* $A_\ell = e(P_\ell, R_\ell)^{k'_\ell}, B_\ell = k_\ell Q_\ell$, $A = e(P, R)^k, B = kQ$, *for random* $k_\ell, k'_\ell, k' \in \mathbb{Z}_q$.

Lemma 76 (Indistinguishability of Valid *vs.* Probe Ciphertexts)**.** *Under the* DBDH *assumption for* $(\mathbb{G}_1, \mathbb{G}_2)$*, given the public key* $\mathsf{params}_{\mathsf{TT}} = (q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ e,\ H,\ P_j,\ Q_j,\ R_j,\ (\alpha_{j,0}R_j,\ \beta_{j,0}P_j,\ \alpha_{j,1}R_j,\ \beta_{j,1}P_j)_{j=1,\ldots,v},\ P,\ Q,\ R,\ (\bar{\alpha}_{j,0}R,\ \bar{\beta}_{j,0}P,\ \bar{\alpha}_{j,1}R,\ \bar{\beta}_{j,1}P)_{j=1,\ldots,v})$ *and the secret keys* $\mathsf{SK}_i \doteq ((\alpha_{j,\omega_j^{(i)}})_{j=1,\ldots,v},\ (\bar{\alpha}_{j,\omega_j^{(i)}})_{j=1,\ldots,v})$ *(*$i \in T$*) it is infeasible to distinguish a valid ciphertext from an ℓ-probe, for any ℓ in the set R_T of undetectable positions for T.*

Sketch. The proof follows from Lemma 68, Definition 5 and from the independence of the "special" sub-schemes from the "shared" sub-schemes in our construction. □

Chapter 7

Scalable Public-Key Tracing and Revoking

7.1 Introduction

An important application of global networking is digital content distribution. In a typical scenario (*e.g.*, Pay-TV) the entities that are active are the *content providers* and the *users* that subscribe to services and receive the content. In addition, the *security manager* is the entity in the system that manages providers and users and is responsible for enforcing various operational rules. For digital content distribution services to remain economically viable in the long run, it is important to design distribution schemes with certain basic properties: (1) strong content-protection—to ensure that only current subscribers have access to the distributed content; (2) traitor-traceability—to counter illegal content reception; (3) transmission efficiency—to optimize the bandwith utilization of the communication medium; and (4) scalability—to support many content providers and a large, dynamically changing population of subscribers. Next, we elaborate on these properties.

Content-protection

Exclusive reception of the digital content can be achieved by employing a multi-user encryption scheme in conjunction with a subscription-based model of service. In this setting only currently subscribed users are able to recover the content successfully. From a security viewpoint, the challenge here is similar to regular encryption systems: to make sure that at any given moment the active subscriber population can access the content whereas outsiders who eavesdrop on the communication medium are incapable of recovering the plaintext content.

Traitor-Traceability

Even if an ideally secure content protection mechanism can be realized, it cannot prevent each subscriber from illicitly share its secret information with non-members. More generally, a group of subscribers (the *traitors*) can collude to construct an illegal decryption device (a *pirate decoder*), which can then be distributed on the black market. We would like to have a mechanism by which misbehaving pirates get caught. One effective such mechanism is provided by the notion of a *traitor tracing scheme*. As discussed in Chapter 3, Section 3.2, 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 fingerprinted. When a pirate decoder is discovered, the security manager can employ a traitor tracing algorithm to uncover the identities of the traitors. The corresponding decryption keys could then be revoked, thus making the pirate decoder useless. A traitor tracing scheme is also a powerful tool against unauthorized access to the content since it allows the uncovering of compromised decryption keys and thus their removal from the system. Ideally, one would like an algorithm able to recover the traitors' identities by just probing the pirate decoder in a *black-box* fashion. However, the inherent hardness of this problem [55] poses limitations on the efficiency attainable with this approach, so that some traitor tracing mechanisms often operate under the assumption that it is possible to extract the actual decryption key that enables the pirate decoder to unscramble the content (*non-black-box* traceability). We will consider both variants in this work.

Transmission efficiency

For efficiency reasons, a distribution scheme should send as few ciphertexts as possible while allowing the legitimate receivers to recover the plaintext content. Also, the amount of users' storage and decryption time should be as small as possible. In an efficient system these parameters must be independent of the total number of user management operations (user additions and removals) as well as of the total number of users.

Scalability

In the context of digital content distribution, scalability has two facets: *server-side* and *client-side*.

Server-side scalability deals with the property that allows the population of content providers to change dynamically. Each content provider in this setting needs access to the encryption mechanism (so that it can scramble content) and access to the distribution channel. Access to the distribution channel can be handled directly using access control mechanisms that are negotiated between the manager of the communication medium and the joining content provider: we will not focus on this aspect; note that in some cases the distribution channel may be a service entirely separated from the subscription service, *e.g.*, when the Internet is used for distribution. On the other hand, access to the encryption mechanism suggests that each content provider needs to have the encryption keys that allow *all users* of the system to get its content. If the content providers are few and closely connected to the security manager, then one may assume that the encryption keys (and perhaps the decryption keys as well) are shared among the providers and the security manager. But this scenario does not scale, since when there are many providers, the amount of keys each user has to store can become prohibitively large (thus violating transmission efficiency property). Therefore, there is a need for providers to use the same key information. If symmetric key methods are being used, a corruption of one provider among the large (sub)-group of providers immediately compromises the content of all providers in that (sub)-group, violating the content protection property. Thus, due to the fact that we deal with a large set of providers and cannot trust all of them, we need an encryption mechanism whose security is not degraded when senders are compromised. This leads to the need of employing public-key cryptography for server-side scalability.

Client-side scalability deals with the fact that we have a user populaion that is changing dynamically due to the service subscription model and security constraints. To allow for a scalable management of user accounts, keys should be easy to generate and revoke. Adversaries that control some user keys that are revoked should be incapable of reading content. We obviously need mechanisms to identify misbehaving users to allow the piracy-deterrence property while the population is dynamically changing.

7.2 Our Results

Given the state of the art, we notice a lack of models and systems where all the properties that constitute a scalable public-key traitor tracing scheme are achieved simultaneously. The study of such a scheme is the undertaking of our work in [38] and our results and approach are outlined in this subsection. An earlier version of this work appeared in [37]; [38] presents detailed modeling of the primitive suggested in this earlier paper, contains all proofs and corrects a flaw in the original construction.

We introduce the first model of a scalable public-key traitor tracing scheme where an unlimited number of users can be added and removed efficiently from the system. Being based on public-key techniques, the scheme supports any number of content providers broadcasting over the same infrastructure. Based on the decisional Diffie-Hellman (DDH) assumption, we then present a concrete scheme meeting these requirements, while preserving the confidentiality of the broadcast from revoked users in the adaptive chosen-plaintext sense. Addition of new users does not affect the keys of users already in the system. Unlimited number of user removals is achieved by dividing the lifetime of the system into *periods*: within a period, a bounded number of user removals can be executed. When an *a priori* specified threshold is reached, a fresh period is started with a New-period operation. For efficiency, such operation does not require private channels between the system manager and the users, and its complexity is independent of the number of users in the system. Within a period, users are not required to maintain state and are stateless. With every New-period operation each user needs to update its private-key information by employing an efficient key-update operation that depends on a security parameter.

The renewal of periods is influenced by the "proactive security model" of Ostrovsky and Yung [68], where information is updated by the manager. Unlike the proactive model, though, each period has a different key, which is reminiscent of the "key insulated" model of security of Dodis *et al.* [40].

In a scalable scheme, adversaries can introduce adversarially-controlled users in the system, they can observe the modifications to the global public key 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 content protection and user-management 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 the two adversaries:

- *Window Adversary*: the adversary obtains the encryption-key as well as some secret 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 an unbounded number of times). Moreover, the adversary is allowed to control arbitrarily many content providers and select the content that is scrambled by the system, except for the challenge broadcast. 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 (specified as a system parameter).

- *Traceability Adversary*: the adversary, as before, obtains some secret keys and constructs a pirate decryption device, employing the secret user-key information (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 (*i.e.*, those contributed to the pirate-key construction). Furthermore, our scheme supports the black-box confirmation method of [17], that even allows a form of traceability in the black-box traitor tracing model.

The advantage of our scalable public-key traitor tracing scheme over previous results comes from the fact that any adversary against the content protection mechanism that is fully revoked in the specified 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 after being 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 [67, 80, 34, 36] possessed this crucial property. Indeed, in all previous public-key schemes, if an adversary, after being revoked, could continue to observe the system operations and cause more user revocations, then she would be able to "revive" her revoked key information and use it to intercept the scrambled content again.

7.3 The Scalable Public-Key Tracing and Revoking Model

In the scalable public-key traitor tracing model, the lifetime of the system is divided into periods. A period is an administrative unit managed based on the system activity and (possibly) on time passing.

A scalable scheme is comprised of the following basic procedures:

- Setup. An initialization procedure that is executed by the security manager. It takes as input a security parameter 1^k and a *saturation limit* 1^v that is an upper bound to the number of users that can be removed within a period. It generates a master secret key MSK along with a public key PK. The security manager keeps MSK secret and publishes PK.

- Add-user. It is a key-generation procedure executed by the security manager. It takes as input the master secret key MSK and the identity i of the new user, and results in a personalized secret key SK_i which is securely communicated to user i.

- Encryption. A public encryption algorithm $\mathcal{E}$ that takes as input the public key PK and a plaintext m, and outputs a ciphertext ψ, to be distributed to the user population through an insecure broadcast channel.

- Decryption. A deterministic algorithm $\mathcal{D}$ that takes as input the secret key SK_i of some user i and a ciphertext ψ, and outputs the corresponding plaintext or the special rejection symbol $\perp$.

- Remove-user. A procedure that given a public key PK, the identity of a user i and the corresponding secret key SK_i, 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_i, while non-removed users should be capable of decrypting it. If the saturation limit has been reached, then a New-period operation has to be executed before removing user i.

- New-period. A procedure executed by the security manager to initiate a fresh period when the saturation limit is reached (a *reactive change*), or when a certain time-limit is reached (a *proactive change*). It takes as input the master secret key MSK and the current public key PK, and results in a new public key PK' and a special reset message to be transmitted over an authenticated but otherwise insecure broadcast channel. Active subscribers can interpret such reset message and update their secret key accordingly; users removed in previous periods, instead, are prevented from doing so by the security properties of the scheme (cf. Section 7.5).

- Tracing. Given the master secret key MSK, the current public key PK and access (either *black-box* or *non-black-box*, cf. Section 7.6) to a pirate decoder, this procedure identifies (at least) one of the traitor users whose keys were employed to construct the pirate decoder.

7.3.1 Scalability Objectives

A scalable scheme should satisfy the following requirements:

- Efficient addition of unlimited number of users throughout the scheme's lifetime. Specifically, the Add-user operation 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 revocation of the decryption capabilities of a set of users within 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 total number of active users in the system, and should only affect the public key of the system.

- Efficient introduction of a new period. The communication overhead for changing a period should be independent of the total number of active users in the system and it should not require private communication channels between the security manager 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).

- 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.

7.3.2 Formal Modeling of Scalable Schemes

A scalable public-key traitor tracing scheme should provide two basic functionalities: on the one hand, the system should be capable of revoking the decryption capabilities of "bad" users; on the other hand, it should be capable of identifying users that participate in the construction of pirate decoders. We formally model the security of revocation and tracing in Section 7.5 and Section 7.6, respectively.

7.4 Construction of a Scalable Public-Key Tracing and Revoking Scheme

Here we present our scheme and show that it is a correct public-key system (*i.e.*, that the public key information allows anyone to encrypt and that a holder of one of the private key representations can apply the decryption algorithm to recover the plaintext).

Setup. The description of a cyclic multiplicative group $\mathbb{G}$ of order q is generated. Then, two random generators $g, g' \in \mathbb{G}$ 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

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

The master secret key is

$$MSK \doteq (A(\cdot), B(\cdot))$$

and the system's public key is

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

where indices $1, \ldots, v$ are used as place-holders. The security manager initiates a new period by publishing PK, and setting the *saturation level* L to 0. L is a system variable known to the security manager.

Add-user. When a new user i requests to join the system, the security manager transmits (over a private channel) the tuple $SK_i \doteq \langle x_i, A(x_i), B(x_i) \rangle$ to user i, where

$$x_i \overset{R}{\leftarrow} \mathbb{Z}_q \quad x_i \notin \{1, \ldots, v\} \cup \mathcal{U}.$$

The set $\mathcal{U}$ is the user-registry containing all values x_i that were selected in previous executions of the Add-user procedure. Subsequently, the security manager 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

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

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

$$\psi \doteq \langle g^r, g'^r, y^r m, \langle z_1, h_1^r \rangle, \ldots, \langle z_v, h_v^r \rangle \rangle.$$

Decryption. The decryption algorithm $\mathcal{D}$ takes as input a secret key $SK_i = \langle x_i, A(x_i), B(x_i) \rangle$ and a ciphertext $\psi = \langle u, u', u'', \langle z_1, u_1 \rangle, \ldots, \langle z_v, u_v \rangle \rangle$. $\mathcal{D}$ first computes the leap-vectors (*cf.* Chapter 2, Section 2.1.3):

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

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 2 (Equations (2.2) and (2.4)), $\vec{\nu}_{A,i}$ and $\vec{\nu}_{B,i}$ agree on all components except for the first: denoting with $(\nu_{A,i})_\ell$ (respectively $(\nu_{B,i})_\ell$) the entry in $\vec{\nu}_{A,i}$ (respectively $\vec{\nu}_{B,i}$) indexed by ℓ, it holds that:

$$\nu_{i,\ell} \doteq (\nu_{A,i})_\ell = (\nu_{B,i})_\ell, for \ell = 1, \ldots, v.$$

The decryption algorithm returns:

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

If ψ is a properly formed ciphertext, *i.e.*

$$\psi = \langle g^r, g'^r, y^r m, \langle z_1, h_1^r\rangle, \ldots, \langle z_v, h_v^r\rangle\rangle$$

then, due to the properties of the leap-vector representation (Equation (2.1)), we have:

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

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:

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

Finally, the saturation level is increased to $L \doteq L + k$.

New-period. When a Remove-user operation is invoked such that the resulting saturation level L would "overflow" the saturation limit v, the security manager starts a new period. First, the security manager broadcasts a special message change period (signed, but not encrypted). Note that we assume that change period is digitally signed by the security manager so that no third parties can maliciously initiate the New-period operation.

Let $enc : \mathbb{Z}_q \to \mathbb{G}$ be an easily invertible encoding that translates a number from $\{0, \ldots, q-1\}$ into an element of $\mathbb{G}$. If $\mathbb{G}$ 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 $enc(a) \in \mathbb{G}$ for any $a \in \mathbb{Z}_q$: this is because $\mathbb{G}$ is the subgroup of quadratic residues modulo p. The encoding function enc can be easily inverted as follows: given $b \doteq enc(a)$, compute the two square roots ρ_1, ρ_2 of a modulo p and define $enc^{-1}(b) = \min\{\rho_1, \rho_2\} - 1$ where min treats ρ_1, ρ_2 as integers in $\{0, \ldots, p-1\}$.

The security manager selects $d_0, \ldots, d_v, e_0, \ldots, e_v$ uniformly at random from $\mathbb{Z}_q$ and transmits the reset message

$$\psi_{\mathsf{reset}} \doteq \langle \mathcal{E}(PK, enc(d_0)), \ldots, \mathcal{E}(PK, enc(d_v)), \\ \mathcal{E}(PK, enc(e_0)), \ldots, \mathcal{E}(PK, enc(e_v)) \rangle$$

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,

$$D(x) = d_0 + d_1 x + \ldots + d_v x^v$$
$$E(x) = e_0 + e_1 x + \ldots + e_v x^v.$$

At this point, the security manager resets the saturation level $L \doteq 0$, updates the two secret polynomials to be:

$$A_{\mathsf{new}}(\cdot) \doteq A(\cdot) + D(\cdot) \pmod q$$
$$B_{\mathsf{new}}(\cdot) \doteq B(\cdot) + E(\cdot) \pmod q$$

and modifies the public key PK as follows:

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

Upon receiving the signed change period message, user i enters a wait-mode. When the user receives the reset message ψ_{reset}, he 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 secret key SK_i to be the new tuple

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

Intuitively, this is secure because a revoked user i will not be able to decrypt any of the ciphertext in the reset message. Therefore, the secret polynomials in the (updated) master secret key will look completely random to user i and his secret key will become useless. A formal security analysis is presented in Section 7.5.

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 $O(v^2)$ in terms of communication), the security manager 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 $\mathbb{G}$). Each non-revoked user will then be able to recover the coefficients $d_0, \ldots, d_v, e_0, \ldots, e_v$ 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 $O(v)$. We omit the details.

7.5 Dealing with Revocation

7.5.1 Model for Revocation

The public-key traitor tracing scheme described in Section 7.4 withstands a more powerful type of attack than what has been considered so far in previous related work [67, 80, 34, 36]. 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 [67, 80, 36].

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 user controlled by the adversary. To reply to such query, a variant of the Add-user operation is executed, in which the adversary is allowed to specify the identity for which she will get the decryption key, (whereas in a regular Add-user operation, the security manager would assign a random identity to the new user). Thus, the Join query models a more powerful adversary that could control the random choice of the security manager. Notice that, after a Join query, the adversary obtains a valid secret key capable of recovering subsequent encrypted broadcasts.

- Revoke query: it models the revocation of a user from the system. To reply to such query, a Remove-user operation is performed and $\mathcal{A}$ is given the new public key that results after the invalidation of the key corresponding to the revoked user.

The main constraint that the above attack scenario imposes on the adversary's behavior is that $\mathcal{A}$ can make at most v Join queries; no restriction is placed on the number of 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.

Note that proving security under this attack scenario does not mean that a real-world implementation of our scheme would withstand only adversaries corrupting up to v users; rather, it provides provable security against adversaries controlling an unlimited number of users, as long as no more than v users are ever corrupted in a row, without the security manager discovering them and revoking their decryption keys within a single period.

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 [67, 80, 34, 36], 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 security manager 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 secret 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 secret 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 secret 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, $\mathcal{A}$ enters 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 (IND-ID-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 IND-ID-CCA1 attacker. Indeed, in our security proof and system design we take advantage of techniques that were developed for dealing with IND-ID-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 $\mathcal{A}$. The game consists of three stages, denoted respectively fst, snd and 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 (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 Revoke queries. At the end of this stage, all the corrupted users get revoked.

The second stage (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 (trd), $\mathcal{A}$ receives a challenge ciphertext ψ^*, 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 $\langle PK, MSK\rangle \overset{R}{\leftarrow} \mathsf{Setup}(1^k, 1^v)$
2. Let $L \doteq 0$, $\mathsf{Corr} \doteq \emptyset$
3. Let $state \doteq \langle L, PK, MSK, \mathsf{Corr}\rangle$
4. $aux \overset{R}{\leftarrow} \mathcal{A}^{\mathsf{Join}(state,\cdot),\mathsf{Revoke}(1,state,\cdot)}(\mathsf{fst}, state.PK)$
5. If $L + |\mathsf{Corr}| > v$ then exit
6. For all $x_j \in \mathsf{Corr}$ do $aux \overset{R}{\leftarrow} aux||\mathsf{Revoke}(0, state, x_j)$
7. $\langle aux, m_0, m_1\rangle \overset{R}{\leftarrow} \mathcal{A}^{\mathsf{Revoke}(1,state,\cdot)}(\mathsf{snd}, aux, state.PK)$
8. $\psi^* \overset{R}{\leftarrow} \mathcal{E}(state.PK, m_{\sigma^*})$, where $\sigma^* \overset{R}{\leftarrow} \{0,1\}$
9. $\sigma \overset{R}{\leftarrow} \mathcal{A}^{\mathsf{Revoke}(state,\cdot)}(\mathsf{trd}, aux, state.PK, \psi^*)$
10. Output 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 exit

(v) set $\mathsf{Corr} \doteq \mathsf{Corr} \cup \{x\}$ and return $(A(x), B(x))$

$\mathsf{Revoke}(\mathsf{isOracle}, state, x)$:

(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 $\mathsf{isOracle} = 1$ and $x \in \mathsf{Corr}$, then 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 $L \doteq L + 1$

(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 reset message ψ_{reset}.

Few points in the above formalization are worth of comment. First, without loss of generality, we assume that the adversary never corrupts the same user twice, as there is no extra information to be gained. We also assume that $\mathcal{A}$ never revokes the users it corrupts, as they get explicitly revoked at Step 6. of the attack game $\mathbf{G}^v_{\mathsf{win}}$.

Second, note that the test at Step 5. is needed to enforce the window constraint: just testing $|\mathsf{Corr}| > v$ would not have been enough to ensure that in Step 6. we can revoke all the corrupted users within the same period.

Finally, note that the code for Revoke is used both as an oracle to $\mathcal{A}$ (Steps 4. and 7.) and as a subroutine for the attack game (Step 6.). To distinguish these two cases, we use the boolean variable $\mathsf{isOracle}$.

Definition 77. *Define $\mathcal{A}$'s advantage as*

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

where the probability is over all the randomness introduced by the window attack game.

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

7.5.2 Security of Revocation

We now formally prove that the scalable public-key traitor tracing scheme described in Section 7.4 is secure against window adversaries (as defined above). In the security proof, we will follow the same structural approach used in [36], first advocated in [32]. Starting from the actual attack scenario, we consider a sequence of hypothetical games, all defined over the same probability space. In each game, the adversary's view is obtained in different ways, but its distribution is still indistinguishable among the games.

The security of our scheme relies on the DDH assumption, as shown below in Theorem 78.

Theorem 78. *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 σ^* used at Step 7. of the attack game $\mathbf{G}^v_{\mathsf{win}}(1^k)$ to create the challenge ciphertext ψ^*: 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). Without loss of generality, 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" New-period operation to be the first one to be caused by the Revoke oracle at Step 7. of the attack game. Depending on the adversary's strategy, such "special" New-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 reset message output by the "special" New-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" New-period operation are totally random, even given all the information in the adversary's view.

In Lemma 80 (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,

$$\left|\Pr[T_1] - \Pr[T_0]\right| \leq (4v + 4)\ \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.1}$$

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:

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

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

$$\Pr[T_2] = \frac{1}{2}. \tag{7.2}$$

In Lemma 81, proven below, we show that the adversary has almost the same chances to guess σ^* in game $\mathbf{G}_1$ and $\mathbf{G}_2$: more precisely,

$$\left|\Pr[T_2] - \Pr[T_1]\right| \leq 2\ \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.3}$$

Combining Equations (7.1), (7.2), and (7.3) together, adversary $\mathcal{A}$'s advantage can be bounded as:

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

□

The core of the proof of Theorem 78 is in the two lemmas that follow, Lemma 80 and Lemma 81.

Overview of the Proof Technique

Throughout the paper, we make extensive use of a technical lemma, stated and proved as Lemma 9 in [32]. For ease of reference, we report it verbatim below.

Lemma 79. *Let k,n be integers with $1 \leq k \leq n$, and let K be a finite field. Consider a probability space with random variables $\vec{\alpha} \in K^{n\times 1}, \vec{\beta} = (\beta_1, \ldots, \beta_k)^T \in K^{k\times 1}, \vec{\gamma} \in K^{k\times 1}$, and $\boldsymbol{M} \in K^{k\times n}$, such that $\vec{\alpha}$ is uniformly distributed over $K^n, \vec{\beta} = \boldsymbol{M}\vec{\alpha} + \vec{\gamma}$, and for $1 \leq i \leq k$, the first i-th rows of $\boldsymbol{M}$ and $\vec{\gamma}$ are determined by $\beta_1, \ldots, \beta_{i-1}$. Then, conditioning on any fixed values of $\beta_1, \ldots, \beta_{k-1}$ such that the resulting matrix $\boldsymbol{M}$ has rank k, the value of β_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 $\vec{V}$ and a vector $\vec{\alpha}$, such that conditioning on any fixed value of $\vec{V}$, $\vec{\alpha}$ 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 $\vec{\beta}$, whose last entry is rand, with the property that fixing a value for $\vec{V}$ and $\vec{\beta}$ also fixes the value of $\vec{\alpha}$, 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 $\vec{\gamma}$) describing the constraints binding vector $\vec{\alpha}$ to vector $\vec{\beta}$, thus obtaining a matrix equation of the form:

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

Finally, we make sure that the preconditions of Lemma 79 are fulfilled; it will follow that the last entry of $\vec{\beta}$ (which is the quantity of interest rand), is distributed uniformly at random in $\mathbb{Z}_q$, even conditioning on fixed values of $\vec{V}$ and of all the other entries of $\vec{\beta}$, 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 Setup algorithm and

$$A^t(\cdot) \doteq A^{t-1}(\cdot) + D^t(\cdot) \quad B^t(\cdot) \doteq B^{t-1}(\cdot) + E^t(\cdot). \tag{7.4}$$

Also define

$$D^{t_1,t_2}(\cdot) \doteq \sum_{t=t_1}^{t_2} D^t(\cdot) \quad E^{t_1,t_2}(\cdot) \doteq \sum_{t=t_1}^{t_2} E^t(\cdot). \tag{7.5}$$

Proofs of Lemmata

Lemma 80. $\left|\Pr[T_1] - \Pr[T_0]\right| \leq (4v+4)\ \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k)$.

Proof. Recall that $\mathbf{G}_1$ differs from $\mathbf{G}_0$ only in the way the 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" 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" 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:

$$\left|\Pr[T_{0,i+1}] - \Pr[T_{0,i}]\right| \leq 2\ \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.6}$$

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" 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:

$$\Pr[T^1_{0,i}] = \Pr[T^0_{0,i}]. \tag{7.7}$$

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" 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,

$$\left|\Pr[T^2_{0,i}] - \Pr[T^1_{0,i}]\right| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.8}$$

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" 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 New-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 New-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 reset message.

As for the "special" 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 $s_1, \ldots, s_i \in \mathbb{G}$, 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 $r^{\hat{t}}_j$ 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 $\mathbf{G}^2_{0,i}$ and $\mathbf{G}^3_{0,i}$. 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 $\mathbf{G}^2_{0,i}$ and $\mathbf{G}^3_{0,i}$, 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

$$\vec{W} \doteq \left(\{c^t_j, r^t_j\}_{\substack{j=1 \\ t \neq \hat{t}}}^{2v+2}, \{c^{\hat{t}}_j, s_j, r^{\hat{t}}_j\}_{j=1}^{i}, \{c^{\hat{t}}_j, r^{\hat{t}}_j\}_{j=i+1}^{2v+2}, r, r' \right)$$

and consider the quantity

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

where Coins represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, σ^* 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 ψ^*.

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 $\vec{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$:

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

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

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

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 $\vec{V}$ and $\vec{\beta}$. In particular, the initial public key PK^0 is fixed by $\vec{\beta}$ and w; the secret keys of the corrupted users are determined by the choice of $\vec{\beta}$, Coins and w; the "special" reset message is fixed by PK^0, Coins, r'' and all the randomness in $\vec{W}$; and the resulting public key $PK^{\hat{t}}$ only depends on PK^0 and $\vec{W}$. Thus, if the distribution of

$\vec{V}$ and $\vec{\beta}$ is the same in both games $\mathbf{G}^2_{0,i}$ and $\mathbf{G}^3_{0,i}$, it will follow that $\Pr[T^3_{0,i}] = \Pr[T^2_{0,i}]$. Since the definition of r'' is the only difference between game $\mathbf{G}^2_{0,i}$ and $\mathbf{G}^3_{0,i}$, and in $\mathbf{G}^3_{0,i}$ 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 $\vec{V}$ and the first $2v+1$ entries of $\vec{\beta}$, is also uniform in $\mathbb{Z}_q$.

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

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

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

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

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 $\vec{\alpha}$ arising from the information in the adversary's view in game $\mathbf{G}^2_{0,i}$ (which, as noted above, can be described just by $\vec{V}$ and $[\vec{\beta}]_2$). In other words, all other constraints on $\vec{\alpha}$ are linear combination of the above, possibly using coefficients from $\vec{V}$. In particular, the constraints that the adversary can derive from knowledge of the values $B^0(x_\ell)$, $\ell = 1, \ldots, v$ (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 $\vec{\alpha}$ 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 v-degree polynomial $A^0(\cdot) + wB^0(\cdot)$, which is completely fixed by the values $\mathbf{X}_0$, $\mathbf{X}_1$, …, $\mathbf{X}_v$, it immediately follows that the constraint on $\vec{\alpha}$ 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 $\mathbf{X}_{z_1}$, …, $\mathbf{X}_{z_v}$ where, for $\ell = 1, \ldots, v$, $\mathbf{X}_{z_\ell}$ is defined as:

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

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 $\vec{\alpha}$ induced by $\mathbf{X}_{z_1}$, …, $\mathbf{X}_{z_v}$ 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 $x_1, \ldots, x_v$ are assumed to be distinct.

As soon as we fix a value for $\vec{V}$, vector $\vec{\gamma}$ and the first $v+1$ rows of matrix $\mathbf{M}$ are completely fixed, but $\vec{\alpha}$ 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 $[\vec{\beta}]_2$, 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 $[\vec{\beta}]_2$), 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 $[\vec{\beta}]_2$ determines the $(v+i+2)$-th row of $\mathbf{M}$, for $i = 1, \ldots, v$. Hence, by Lemma 79, we can conclude that the conditional distribution of $[r'']_2$, with respect to $\vec{V}$ and to all other components of $[\vec{\beta}]_2$, is also uniform over $\mathbb{Z}_q$, from which it follows that

$$\Pr[T^3_{0,i}] = \Pr[T^2_{0,i}]. \tag{7.9}$$

Game $\mathbf{G}^4_{0,i}$. Game $\mathbf{G}^4_{0,i}$ is defined to be identical to $\mathbf{G}_{0,i+1}$. Thus, $\mathbf{G}^4_{0,i}$ differs from $\mathbf{G}^3_{0,i}$ only in that the values u and u' in the $(i+1)$-th ciphertext in the "special" reset message are consistent, rather than being nearly independent, as in game $\mathbf{G}^3_{0,i}$. 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 $\mathbf{G}^3_{0,i}$ and $\mathbf{G}^4_{0,i}$ can be used to distinguish Diffie-Hellman tuples from totally random tuples. Hence,

$$\left|\Pr[T^4_{0,i}] - \Pr[T^3_{0,i}]\right| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.10}$$

Combining Equations (7.7), (7.8), (7.9) and (7.10) we get Equation (7.6), 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. $\square$

Lemma 81. $\left|\Pr[T_2] - \Pr[T_1]\right| \leq 2\,\mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k)$.

Proof. Recall that $\mathbf{G}_2$ differs from $\mathbf{G}_1$ only in the way the challenge ciphertext ψ^* is computed: in particular, in game $\mathbf{G}_1$, ψ^* encrypts either one of the two messages m_0 and m_1 chosen by the adversary, whereas in $\mathbf{G}_2$, ψ^* encrypts a totally random message $m \overset{R}{\leftarrow} \mathbb{G}$.

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

$$\Pr[T_1^1] = \Pr[T_1^0]. \tag{7.11}$$

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,

$$\left| \Pr[T_1^2] - \Pr[T_1^1] \right| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.12}$$

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 $\textsf{New-period}$ operation before the challenge ciphertext was created. (If no $\textsf{New-period}$ 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 $\textsf{New-period}$ operations that occured during the entire game, and for $t = 1, \ldots, \bar{t}$, let c_1^t, ..., c_{2v+2}^t be the coefficients of the randomizing polynomials $D^t(\cdot)$ and $E^t(\cdot)$ used in the t-th $\textsf{New-period}$ 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 $\textsf{reset}$ message.

Observe that the challenge ciphertext ψ^* 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" $\textsf{New-period}$ operation to occur or not. The case in which no $\textsf{New-period}$ operation occurred at Step 7. of the attack game is slightly simpler, so we discuss it first.

Case 1. Consider the quantity

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

where $\textsf{Coins}$ represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, and σ^* 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 $\vec{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$:

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

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

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

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 $\vec{V}$ and $\vec{\beta}$. Thus, if the distribution of $\vec{V}$ and $\vec{\beta}$ 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 $\vec{V}$ and the first $2v+1$ entries of $\vec{\beta}$, is also uniform in $\mathbb{Z}_q$.

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

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

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

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

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

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

The above matrix $\mathbf{M}$ 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 $\vec{\alpha}$, 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 ψ^* are exactly those of the users corrupted by the adversary. Hence, the constraints on $\vec{\alpha}$ arising from the last v components of the challenge ciphertext ψ^* can be obtained as linear combination of the constraints specified by $\mathbf{M}$.

As soon as we fix a value for $\vec{V}$, the first $2v+1$ entries of vector $\vec{\gamma}$ and the first $v+1$ rows of matrix $\mathbf{M}$ are completely fixed, but $\vec{\alpha}$ 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 $[\vec{\beta}]_2$, 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 $[\vec{\beta}]_2$), 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 $[\vec{\beta}]_2$ determines the $(v+\ell+2)$-th row of $\mathbf{M}$, for $\ell = 1, \dots, v$. In particular, fixing all the entries of $[\vec{\beta}]_2$ 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 $\vec{\gamma}$ is fixed, too. Hence, by Lemma 79, we can conclude that the conditional distribution of $[r''^*]_2$, with respect to $\vec{V}$ and to all the other components of $[\vec{\beta}]_2$, is also uniform over $\mathbb{Z}_q$, from which it follows that

$$\Pr[T_1^3] = \Pr[T_1^2]. \tag{7.13}$$

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

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

where Coins represents the coin tosses of $\mathcal{A}$, $w \doteq \log_g g'$, σ^* 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 $\mathbb{G}$ 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 $\vec{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(4v+4)\times 1}$, defined as:

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

Consider the vector $\vec{\beta} \in \mathbb{Z}_q^{(4v+3)\times 1}$ defined as

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

It is clear by inspection that all the information in the adversary's view is completely determined by $\vec{V}$ and $\vec{\beta}$. In particular, the initial public key PK^0 is fixed by $\vec{\beta}$ and w; the secret keys of the corrupted users are determined by the choice of $\vec{\beta}$, Coins and w; the "special" reset message is fixed by PK^0, Coins, and all the randomness in $\vec{V}$; the resulting public key $PK^{\hat{t}}$ only depends on $\vec{\beta}$ and w; and the adversary's choice of m_0 and m_1 is fixed by $\vec{V}$ and the first $4v+2$ entries of $\vec{\beta}$.

Thus, if the distribution of $\vec{V}$ and $\vec{\beta}$ 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 $\vec{V}$ and the first $4v+2$ entries of $\vec{\beta}$, is also uniform in $\mathbb{Z}_q$.

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

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

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

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

and $\mathbf{M} \in \mathbb{Z}_q^{(4v+3)\times(4v+4)}$ is defined as:

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

The matrix $\mathbf{M}$ describes all the constraints on the $(4v+4)$ unknowns represented by $\vec{\alpha}$, 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 $\vec{\alpha}$ arising from the last v components of the challenge ciphertext ψ^*. 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 $z_1^*, \dots, z_v^*$ specified in the public key PK^{t^*} that is used to create the challenge ciphertext are all different from the identities $x_1, \dots, x_v$ 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 $\vec{V}$, the first $4v + 2$ entries of vector $\vec{\gamma}$ and the first $2v + 2$ rows of matrix $\mathbf{M}$ are completely fixed, but $\vec{\alpha}$ 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 $[\vec{\beta}]_2$, 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 $[\vec{\beta}]_2$), 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 $[\vec{\beta}]_2$ determines the $(2v+\ell+3)$-th row of $\mathbf{M}$, for $\ell = 1, \ldots, v$. In particular, fixing the first $3v+2$ entries of $[\vec{\beta}]_2$ 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 $\vec{\gamma}$ get fixed, too. Hence, by Lemma 79, we can conclude that the conditional distribution of $[r''^*]_2$, with respect to $\vec{V}$ and to all other components of $[\vec{\beta}]_2$, is uniform over $\mathbb{Z}_q$, from which it follows that Equation (7.13) 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 ψ^* 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,

$$\left|\Pr[T_1^4] - \Pr[T_1^3]\right| \le \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.14}$$

Combining Equations (7.11), (7.12), (7.13) and (7.14), the thesis follows. □

7.6 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" 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 7.4.

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 c suspected users,[1] checks whether the suspected set includes all the secret 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 c users and deterministically recovers the identities of all the traitors.

7.6.1 Model for Traceability

The traceability adversary operates similarly to the window adversary described in Section 7.5. Namely, after receiving the initial public key of the system, adversary $\mathcal{A}$ can interleave (in any adaptively-chosen order) up to c Join queries, upon which $\mathcal{A}$ receives the secret keys of the corresponding users (the *traitors*), and a polynomial number of Revoke queries. Notice that each 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 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

The above attack scenario can be formalized in terms of the following traceability adversary attack game $\mathbf{G}^{c}_{\mathsf{trt}}(1^k)$, played between a challenger and the adversary.

1. Let $\langle PK, MSK\rangle \doteq \mathsf{Setup}(1^k, 1^c)$

2. Let $L \doteq 0$, $\mathcal{T} \doteq \emptyset$

[1]Recall, c denotes the *collusion* threshold, and should not be confused with the *revocation* threshold v defined in Section 7.4; *e.g.*, in our schemes $c = \lfloor \frac{v}{2} \rfloor$.

3. Let $state \doteq \langle L, PK, MSK, \mathcal{T} \rangle$

4. $\mathsf{D} \overset{R}{\leftarrow} \mathcal{A}^{\mathsf{Join}(state,\cdot),\mathsf{Revoke}(1,state,\cdot)}(state.PK)$

5. If $|\mathcal{T}| > c$ 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 7.5.1, except that the role of the set Corr is now played by the set $\mathcal{T}$.

Definition 82. *For any public key PK, define the success probability of a decoder D as:*

$$\mathsf{Succ}_{PK}(\mathsf{D}) \doteq \Pr[m' = m \mid \; m \overset{R}{\leftarrow} \mathbb{G}; \psi^* \overset{R}{\leftarrow} \mathcal{E}(PK, m); m' \overset{R}{\leftarrow} \mathsf{D}(\psi^*)]$$

where the probability is over the random choice of m, the randomness used to create the challenge ciphertext ψ^* 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) c 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 Sections 7.6.2 and 7.6.3, respectively.

7.6.2 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 identify (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 [17, 67, 80] 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 c 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 c-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 [55], 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 ε-fraction of correctly formed messages (rather than with probability 1), where ε is the desired threshold below which the decoder is considered "useless" (following the "threshold tracing" approach of [66]).

Definition 83 (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 84 (ε-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* $\boldsymbol{G}^{c}_{\mathsf{trt}}(1^k)$ *with the challenger. A Black-Box Confirmation algorithm* BBC *satisfies* ε-Black-Box

Confirmation *if, for any probabilistic-polynomial time adversary* $\mathcal{A}$ *playing the* $\boldsymbol{G}^{\mathsf{c}}_{\mathsf{trt}}(1^k)$ *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 ε/c). 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 $\varepsilon/2c$), it proclaims i as a traitor. Otherwise, it sets $I \doteq I \setminus \{i\}$, 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 7.4 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 $\mathsf{New\text{-}period}$ operations that occur during the entire game, and for $t = 1, \ldots, \bar{t}$, let c_1^t, …, 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 reset message. By Equation (7.4), $(A^{\bar{t}}(\cdot), B^{\bar{t}}(\cdot))$ 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 $|I| \leq c \leq v/2$, 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 $y' \doteq g^{A'(0)} g'^{B'(0)}$, and $h'_\ell \doteq g^{A'(z_\ell)} g'^{B'(z_\ell)}$, for $\ell = 1, \ldots, v$.

Correctness of Black-Box Tracing

The correctness of the black-box tracing algorithm described above follows from Theorem 85 and Theorem 87 stated below. Theorem 85 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.* $\delta(\mathsf{Susp}) \gtrsim \varepsilon$.[2] 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 $\varepsilon/2c$) 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 87 below), i must be one of the traitors. This shows the confirmation property.

Theorem 85. *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 ψ^* to the pirate decoder D; different games only differs in the way ψ^* is computed. In the original game

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

$\mathbf{G}_0$, 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 ψ^*; for each game $\mathbf{G}_j$, let T_j be the event that $m = m'$ in $\mathbf{G}_j$.

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

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

By definition, we have that

$$\Pr[T_0] = \mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D}). \tag{7.15}$$

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:

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

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

$$\Pr[T_1] = \Pr[T_0]. \tag{7.16}$$

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

$$E1'. \quad r \xleftarrow{R} \mathbb{Z}_q; \quad r' \xleftarrow{R} \mathbb{Z}_q \setminus \{r\}$$

$$E3'. \quad u' \leftarrow g'^{r'}$$

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 probabilistic-polynomial time adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\left| \Pr[T_2] - \Pr[T_1] \right| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.17}$$

Game $\mathbf{G}_3$: To turn game $\mathbf{G}_2$ into game $\mathbf{G}_3$, we consider the set 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

$$A'(x_s) = A^{\bar{t}}(x_s) \quad B'(x_s) = B^{\bar{t}}(x_s), \ \forall s \in \mathsf{Susp} \tag{7.18}$$

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:

$$E4''. \quad u'' \leftarrow u^{A'(0)} u'^{B'(0)} m$$

$$E5''. \quad u_\ell \leftarrow u^{A'(z_\ell)} u'^{B'(z_\ell)}, \quad \ell = 1, \ldots, v$$

Using the technique outlined in Section 7.5.2, in Lemma 86 below, we show that

$$\Pr[T_3] = \Pr[T_2]. \tag{7.19}$$

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:

$$E1''. \quad r \xleftarrow{R} \mathbb{Z}_q$$

$$E3''. \quad u' \leftarrow g'^{r}$$

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 probabilistic-polynomial time adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\left|\Pr[T_4] - \Pr[T_3]\right| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.20}$$

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.*,

$$\Pr[T_4] = \delta(\mathsf{Susp}). \tag{7.21}$$

Combining Equations (7.15), (7.16), (7.17), (7.19), (7.20) and (7.21), 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:

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

□

Lemma 86. $\Pr[T_3] = \Pr[T_2]$.

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:

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

where $\mathsf{Coins}_{\mathcal{A}}$ denotes the coin tosses of $\mathcal{A}$, $\mathsf{Coins}_{\mathsf{D}}$ denotes the coin tosses of D, $w \doteq \log_g g'$, m is the random message encrypted within ψ^*, r and r' are the random values used to create ψ^*, 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 $\mathbf{G}^c_{\mathsf{trt}}(1^k)$ 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 Setup algorithm in Step 1. of the $\mathbf{G}^c_{\mathsf{trt}}(1^k)$ attack game) and $A'(\cdot)$, $B'(\cdot)$ (used in game $\mathbf{G}_3$). This randomness can be represented with the vector

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

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

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

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, subject to the constraints arising from imposing Equation (7.18).

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

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

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

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

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

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 $\mathbf{G}^c_{\mathsf{trt}}(1^k)$ is completely determined by $\vec{V}$ and $\vec{\vec{\beta}}$. In particular, the initial public key PK^0 is fixed by $\vec{\vec{\beta}}$ and w, and the secret keys of the traitors are determined by the choice of $\vec{\vec{\beta}}$, $\mathsf{Coins}_{\mathcal{A}}$ and w.

Besides the information in $\mathcal{A}$'s view, which is completely determined by the value of $\vec{V}$ and $\vec{\vec{\beta}}$, the only other quantity affecting D's behavior is the ciphertext ψ^*. 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 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 $\vec{V}$, $\vec{\bar{\beta}}$ 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 $\vec{V}$, $\vec{\bar{\beta}}$ and by the v-degree polynomial $X'(\cdot) \doteq rA'(\cdot) + wr'B'(\cdot)$. Moreover, $[\psi^*]_2$ depends on $\vec{V}$, $\vec{\bar{\beta}}$ and $X^{\bar{t}}(\cdot)$ according to the same functional dependence of $[\psi^*]_3$ upon $\vec{V}$, $\vec{\bar{\beta}}$ and $X'(\cdot)$. Therefore, to prove the lemma, it suffices to show that, conditioning on any fixed values of $\vec{V}$ and $\vec{\bar{\beta}}$, $X^{\bar{t}}(\cdot)$ 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 value at x_{t_j} is $r\mathbf{A}_j + wr'\mathbf{B}_j$, for $j = 1, \ldots, c$.

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

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

and

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

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

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

and

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

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

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

Therefore, it follows that

$$\mathbf{X}_{v-c+j} = \mathbf{X}'_{v-c+j},\ j = 1, \ldots, c. \tag{7.22}$$

It only remains to be proven that, conditioning on fixed values of $\vec{V}$ and $\vec{\bar{\beta}}$, the tuple $\mathbf{X}_0^{\bar{t}}$, …, $\mathbf{X}_{v-c}^{\bar{t}}$ and the tuple $\mathbf{X}'_0$, …, $\mathbf{X}'_{v-c}$ are distributed according to the same joint conditional distribution. (Notice that fixing a value for $\vec{V}$ and $\vec{\vec{\beta}}$, immediately fixes a value for the tuple $\mathbf{X}^{\bar{t}}_{v-c+j}$, which by Equation (7.22) is equal to $\mathbf{X}'_{v-c+j}$, $j = 1, \ldots, c$.)

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 Equation (7.18). It follows that 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 $\vec{V}$ and $\vec{\vec{\beta}}$, the tuple $\mathbf{X}'_0, \ldots, \mathbf{X}'_{v-c}$ is distributed uniformly at random in $\mathbb{Z}_q^{(v-c+1)\times 1}$. Hence, it suffices to show that, for $\ell = 0$, $\ldots$, $v-c$, the conditional distribution of $\mathbf{X}^{\bar{t}}_\ell$ with respect to $\vec{V}$, $\vec{\vec{\beta}}$ and $\mathbf{X}^{\bar{t}}_0, \ldots, \mathbf{X}^{\bar{t}}_{\ell-1}$ is uniform over $\mathbb{Z}_q$. To this aim, fix $\ell \in \{0, \ldots, v-c\}$, and consider the following matrix equation:

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

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

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

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

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

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

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

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 $v + c + \ell + 2$. As soon as we fix $\vec{V}$, vector $\vec{\gamma}_\ell$ and the first $v + 1$ rows of $\mathbf{M}_\ell$ are determined, but $\vec{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. Similarly to the proof of Lemma 81, it is also possible to show that fixing the first $v+j$ entries of $\vec{\vec{\beta}}$ determines the $(v+j+1)$-th row of $\mathbf{M}_\ell$, for $j = 1, \dots, c$; and that moreover, fixing the first $v+c+1$ entries of $\vec{\vec{\beta}}$ determines all the remaining rows of $\mathbf{M}_\ell$.

Hence, by Lemma 79, we can conclude that the conditional distribution of $\mathbf{X}_\ell^{\bar{t}}$ with respect to ($\vec{V}$, $\vec{\vec{\beta}}$, $\mathbf{X}_0^{\bar{t}}$, ..., $\mathbf{X}_{\ell-1}^{\bar{t}}$) is uniform over $\mathbb{Z}_q$, $\forall \ell \in \{0, \dots, v-c\}$. 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, $(\vec{V}, \vec{\vec{\beta}}, X^{\bar{t}}(\cdot)))$ has the same joint distribution as ($\vec{V}$, $\vec{\vec{\beta}}$, $X'(\cdot)$), 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 87. *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 85, we define a sequence of "indistinguishable" games $\mathbf{G}_0$, $\mathbf{G}_1$, $\ldots$: 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:

$$A'(x_s) = A^{\bar{t}}(x_s) \quad B'(x_s) = B^{\bar{t}}(x_s),\ \forall s \in I. \tag{7.23}$$

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

$E1.\quad r \xleftarrow{R} \mathbb{Z}_q$

$E2.\quad u \leftarrow g^r$

$E3.\quad u' \leftarrow g'^r$

$E4.\quad u'' \leftarrow g^{A'(0)r} g'^{B'(0)r} m$

$E5.\quad u_\ell \leftarrow g^{A'(z_\ell)r} g'^{B'(z_\ell)r}, \quad \ell = 1, \ldots, v$

$E6.\quad \psi^* \leftarrow \langle u, u', u'', \langle z_1, u_1 \rangle, \ldots, \langle z_v, u_v \rangle \rangle$

By definition, we have that:

$$\Pr[T_0] = \delta(I). \tag{7.24}$$

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:

$E4'.\quad u'' \leftarrow u^{A'(0)} u'^{B'(0)} m$

$E5'.\quad u_\ell \leftarrow u^{A'(z_\ell)} u'^{B'(z_\ell)}, \quad \ell = 1, \ldots, v$

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

$$\Pr[T_1] = \Pr[T_0]. \tag{7.25}$$

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:

$E1'.\quad r \xleftarrow{R} \mathbb{Z}_q; \quad r' \xleftarrow{R} \mathbb{Z}_q \setminus \{r\}$

$E3'.\quad u' \leftarrow g'^{r'}$

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 probabilistic-polynomial time adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\big|\Pr[T_2] - \Pr[T_1]\big| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.26}$$

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

$$A''(x_s) = A^{\bar{t}}(x_s) \quad B''(x_s) = B^{\bar{t}}(x_s),\ \forall s \in I \setminus \{i\} \tag{7.27}$$

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

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

Notice that, by Equations (7.23) and (7.27), it holds that

$$A''(x_s) = A'(x_s) \quad B''(x_s) = B'(x_s),\ \forall s \in I \setminus \{i\}.$$

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

$E4''$. $u'' \leftarrow u^{A''(0)}u'^{B''(0)}m$

$E5''$. $u_\ell \leftarrow u^{A''(z_\ell)}u'^{B''(z_\ell)}, \quad \ell = 1, \ldots, v$

Using the technique described in Section 7.5.2, in Lemma 88 below, we show that

$$\Pr[T_3] = \Pr[T_2]. \tag{7.28}$$

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:

$E1''$. $r \xleftarrow{R} \mathbb{Z}_q$

$E3''$. $u' \leftarrow g'^r$

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 probabilistic-polynomial time adversary able to distinguish Diffie-Hellman tuples from totally random tuples with non-negligible advantage. Hence,

$$\left|\Pr[T_4] - \Pr[T_3]\right| \leq \ \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(k). \tag{7.29}$$

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.*,

$$\Pr[T_4] = \delta(I \setminus \{i\}). \tag{7.30}$$

By Equations (7.24), (7.25), (7.26), (7.28), (7.29) and (7.30), 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:

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

□

Lemma 88. $\Pr[T_3] = \Pr[T_2]$

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 $\bar{c} \doteq |\mathcal{T} \cap I|$, where $\mathcal{T} = \{t_1, \ldots, t_c\}$ is the set of traitors; without loss of generality assume that $\mathcal{T} \cap I = \{t_1, \ldots, t_{\bar{c}}\}$. Also, set

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

Notice that, since $i \notin \mathcal{T}$, for $1 \leq j \leq \bar{c}$ 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:

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

where $\mathsf{Coins}_{\mathcal{A}}$ denotes the coin tosses of $\mathcal{A}$, $\mathsf{Coins}_{\mathsf{D}}$ denotes the coin tosses of 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 ψ^*, r and r' are the random values used to create ψ^*, 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 $\mathbf{G}^c_{\mathsf{trt}}(1^k)$.

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 Setup algorithm in Step 1. of the $\mathbf{G}^c_{\mathsf{trt}}(1^k)$ 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

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

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

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

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, subject to the constraints arising from imposing Equation (7.23), and the vector

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

which is uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$, subject to the constraints arising from imposing Equation (7.27).

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

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

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 $\mathbf{G}^c_{\mathsf{trt}}(1^k)$ is completely determined by $\vec{V}$ and $\vec{\bar{\beta}}$. In particular, the initial public key PK^0 is fixed by $\vec{V}$, and the secret keys of the traitors are determined by the choice of $\vec{\bar{\beta}}$, $\mathsf{Coins}_{\mathcal{A}}$ and w.

Besides the information in $\mathcal{A}$'s view, the only other quantity affecting D's behavior is the ciphertext ψ^*. 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 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 $\vec{V}$, $\vec{\bar{\beta}}$ 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 $\vec{V}$, $\vec{\bar{\beta}}$ and by the v-degree polynomial $X''(\cdot) \doteq rA''(\cdot) + wr'B''(\cdot)$. Moreover, $[\psi^*]_2$ depends on $\vec{V}$, $\vec{\bar{\beta}}$ and $X'(\cdot)$ according to the same functional dependence of $[\psi^*]_3$ upon $\vec{V}$, $\vec{\bar{\beta}}$ and $X''(\cdot)$. Therefore, to prove the Lemma, it suffices to show that, conditioning on any fixed values of $\vec{V}$ and $\vec{\bar{\beta}}$, $X'(\cdot)$ 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 Equation (7.23). It follows that 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 Equation (7.27). It follows that 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 with respect to $\vec{V}$ and $\vec{\bar{\beta}}$, it suffices to show that, conditioning on any fixed values of $\vec{V}$ and $\vec{\bar{\beta}}$, the value $\mathbf{X}'_i$ is distributed uniformly at random in $\mathbb{Z}_q$.

To this aim, consider the following matrix equation:

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

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

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

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

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

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

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

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 $v + \bar{c} + 2$. As soon as we fix $\vec{V}$, vector $\vec{\gamma}$ and the first $v+1$ rows of $\mathbf{M}$ are determined, but $\vec{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. Similarly to the proof of Lemma 81, it is also possible to show that fixing the first $v+j$ entries of $\vec{\vec{\beta}}$ determines the $(v+j+1)$-th row of $\mathbf{M}$, for $j = 1, \ldots, \bar{c}$; and that moreover, fixing the first $v + \bar{c} + 1$ entries of $\vec{\vec{\beta}}$ determines all the remaining rows of $\mathbf{M}$.

By Lemma 79, we can conclude that the conditional distribution of $\mathbf{X}'_i$ with respect to $\vec{V}$ and $\vec{\vec{\beta}}$ is uniform over $\mathbb{Z}_q$. In other words, conditioning on the information seen by the adversary before receiving the challenge ψ^*, the value of $X'(\cdot)$ at x_i looks random over $\mathbb{Z}_q$. Thus, $(\vec{V}, \vec{\vec{\beta}}, X'(\cdot))$ has the same joint distribution as $(\vec{V},\ \vec{\vec{\beta}},\ X''(\cdot))$, completing the proof. $\square$

7.6.3 Non-Black-Box Tracing

In Section 7.6.3 we describe a non-black-box tracing algorithm which builds on the results of [17, 67], but it is tailored to our family of representations. Then, in Section 7.6.3, we analyze its security properties in the formal model for traceability of Section 7.6.1, under a non-black-box assumption, given below as Assumption 1. Before that, however, we develop some notation.

Notation

Recall that in the scheme of Section 7.4, 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 Chapter 2, Definition 2) two leap-vectors:

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

where $(A(\cdot), B(\cdot))$ is the master secret key corresponding to the current public key PK. By Equations (2.2) and (2.4), $\vec{\nu}_{A,i}$ and $\vec{\nu}_{B,i}$ agree on the last v components; thus, under the current public key PK, user x_i's secret key can be compactly rewritten as

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

where $\lambda_0^{(i)}$, $\lambda_1^{(i)}, \ldots, \lambda_v^{(i)}$ are the Lagrange coefficients defined in Equations (2.3) and (2.4). (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 $\vec{\delta}_i$ is a representation of y with respect to g, g', h_1, …, h_v; for short, when this is the case, in the following we will just say that $\vec{\delta}_i$ is a valid representation of the public key PK. Also notice that *any* such valid representation $\vec{\delta}$ of the current public key PK would work for decrypting messages encrypted with PK; for a generic valid representation

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

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

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

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 1 (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 $\boldsymbol{G}^{\mathsf{c}}_{\mathsf{trt}}(1^k)$ with the challenger. If D can correctly decrypt random ciphertexts encrypted using $PK_{\mathcal{A}}$ (in other words, $\mathsf{Succ}_{PK_{\mathcal{A}}}(\mathsf{D}) = 1$), then D contains a valid representation $\vec{\delta}$ of $PK_{\mathcal{A}}$, and it is possible to reverse-engineer D and extract $\vec{\delta}$.

Assumption 1 is partially supported by Proposition 3 and it is essentially equivalent to what was previously assumed in [17]. It is also *a priori* much less restrictive than the non-black-box assumption made in [67], 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 89 (whose proof is given in Section 7.6.3) we show that in our context, the seemingly more restrictive assumption from [67] actually follows from Assumption 1 and Assumption 10.

Lemma 89. *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 $\boldsymbol{G}^{c}_{\mathsf{trt}}(1^k)$ with the challenger. Also let $\mathcal{T} \doteq \{t_1, \ldots, t_c\}$ and, for $j = 1, \ldots, c$, denote with $\vec{\delta}_{t_j}$ the compact representation of the secret key of user t_j with respect to the public key $PK_{\mathcal{A}}$. If the pirate decoder D output by $\mathcal{A}$ contains a valid representation $\vec{\delta}$ for the public key $PK_{\mathcal{A}}$, such that $\vec{\delta}'$ is not a linear combination of $\vec{\delta}'_{t_1}, \ldots, \vec{\delta}'_{t_c}$, then the discrete logarithm problem over $\mathbb{G}$ is solvable.*

Non-Black-Box Tracing Algorithm

We present a deterministic tracing algorithm that recovers, under Assumptions 10 and 1, the identities of the traitors that created the pirate key. Suppose that the content of a pirate decoder is exposed. By Assumption 1, it is possible to extract from D a valid representation $\vec{\delta}$ 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 $\vec{\delta_1}, \ldots, \vec{\delta_n}$ 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.[3] Remember that the secret key of user j with respect to the current public key can be compactly represented in the form

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

[3]Without loss of generality we are assuming that the current saturation level L is equal to v.

where $\lambda_j^{(j)}, \lambda_{i_1}^{(j)}, \ldots, \lambda_{i_v}^{(j)}$ are the Lagrange coefficients defined in Equations (2.3) and (2.4). Notice that, for any polynomial $P(\cdot) \in \mathbb{Z}_q^v[x]$, 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 jth row is $\vec{\delta'_j}$, for $j = 1, \ldots, n$, *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 $\{t_1, \ldots, t_c\} \subseteq \{1, \ldots, n\}$. By Lemma 89 and Assumption 10, $\vec{\delta'}$ must be a linear combination of the vectors $\vec{\delta'_{t_1}}, \ldots, \vec{\delta'_{t_c}}$ obtained by projecting the traitors' secret keys $\vec{\delta_{t_1}}, \ldots, \vec{\delta_{t_c}}$ onto the last v components. It follows that $\vec{\delta'}$ also lies in the linear span of $\vec{\delta'_1}, \ldots, \vec{\delta'_n}$. More precisely, there exists a vector $\vec{\varphi}$ of Hamming weight at most c such that

$$\vec{\delta'} = \vec{\varphi} \cdot \mathbf{A}. \tag{7.31}$$

Consider the two matrices:

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

It is easy to verify that $\mathbf{A} \cdot \mathbf{B} = \mathbf{H}$. Multiplying Equation (7.31) by $\mathbf{B}$, we get

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

where

$$\vec{\delta''} \doteq \vec{\delta'} \cdot \mathbf{B}. \tag{7.32}$$

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

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

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

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

In Lemma 90 (Section 7.6.3), 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.* [63]. Generalized Reed-Solomon Codes can be decoded efficiently by the algorithm of Berlekamp and Welch [11]. This means that, for any $e \leq c$ and any vector $\vec{\mu} \in \mathbb{Z}_q^n$, there exists (at most) a unique vector $\vec{\omega} \in \mathcal{C}$ that disagrees with $\vec{\mu}$ in at most e positions (since $\mathcal{C}$ has distance $(v+1)$ and $c = \lfloor \frac{v}{2} \rfloor$). Moreover, such unique vector $\vec{\omega} \in \mathcal{C}$ (if it exists) can be recovered in deterministic polynomial-time. We now describe how this can be exploited to reconstruct $\vec{\varphi}$ given $\vec{\delta'}$.

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

$$\vec{\vartheta} \cdot \mathbf{H} = \vec{\delta''}. \tag{7.33}$$

where $\vec{\delta''}$ is defined in Equation (7.32). Note that such $\vec{\vartheta}$ can be found by standard linear algebra since Equation (7.33) 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

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

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

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

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

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

By Equation (7.31), $\vec{\varphi}$ is a vector of Hamming weight at most c, whose non-zero components correspond to the identities of the traitors; thus, the traitors' identities can be recovered as

$$\{t_1, \ldots, t_c\} \doteq \{j \in \{1, \ldots, n\} \mid \vec{\varphi}_j \neq 0\}.$$

Time-Complexity. The tracing procedure has time complexity $O(n^2)$, which can be optimized to $O(n(\log n)^2)$, if matrix operations are implemented in a more sophisticated manner, see *e.g.* [12]. If the number of traitors exceeds the bound c, it is still possible to extract candidate sets of potential traitors using the Guruswami-Sudan algorithm [50], 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 89 and 90, 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 89

Let g be a generator of $\mathbb{G}$, and let $g' \doteq g^w$. Using adversary $\mathcal{A}$ described in the attack game $\mathbf{G}^c_{\mathsf{trt}}(1^k)$, we want to show how to recover the value w. In performing Step 1. of $\mathbf{G}^c_{\mathsf{trt}}(1^k)$, choose two random polynomials $A^0(\cdot)$ and $B^0(\cdot)$ 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 7.6.1; in particular, let $\bar{t}$ be the number of $\mathsf{New\text{-}period}$ operation occurring during the entire game. Eventually, adversary $\mathcal{A}$ outputs a pirate decoder D from which (by Assumption 1) it is possible to extract a vector

$$\vec{\delta} = \langle \gamma_a, \gamma_b, \gamma_1, \ldots, \gamma_v \rangle,$$

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

$$y = g^{\gamma_a} g'^{\gamma_b} \prod_{\ell=1}^{v} h_\ell^{\gamma_\ell} \tag{7.34}$$

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 (7.34), 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:

$$w\Big(\gamma_b + \sum_{\ell=1}^{v} B^{\bar{t}}(x_{i_\ell})\gamma_\ell - B^{\bar{t}}(0)\Big) = A^{\bar{t}}(0) - \gamma_a - \sum_{\ell=1}^{v} A^{\bar{t}}(x_{i_\ell})\gamma_\ell \tag{7.35}$$

Notice that both the right-hand side and the coefficient of w in Equation (7.35) are known, so that if such coefficient is non-zero (or, equivalently, if the right-hand side of Equation (7.35) is non-zero), then we can successfully recover the value of w, thus violating Assumption 10. To complete the argument, it then suffices to show that the right-hand side of Equation (7.35) is zero only with negligible probability, or equivalently that:

$$\Pr[\gamma_a = \bar{\gamma}_a] = 1/q \tag{7.36}$$

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 γ_a equal to $\bar{\gamma}_a$ are just 1 in q, proving Equation (7.35) and thus the lemma.

To prove that $\bar{\gamma}_a$ is distributed uniformly in $\mathbb{Z}_q$, we again make use of Lemma 79 following the same approach described in Section 7.5.2.

Consider the quantity

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

where 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 $\mathbf{G}_{\mathsf{trt}}^{c}(1^k)$ 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 $\vec{\alpha}$ uniformly distributed in $\mathbb{Z}_q^{(2v+2)\times 1}$:

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

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

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

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 $\mathbf{G}_{\mathsf{trt}}^{c}(1^k)$ is completely determined by $\vec{V}$ and $\vec{\beta}$. In particular, the initial public key PK^0 is fixed by $\vec{\beta}$ and w, and the secret keys of the traitors are determined by the choice of $\vec{\beta}$, Coins and w.

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

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

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

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

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

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

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

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

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

As soon as we fix $\vec{V}$, the first $v+c+1$ entries of $\vec{\gamma}$ and the first $v+1$ rows of $\mathbf{M}$ are determined, but $\vec{\alpha}$ is still distributed uniformly and independently at random in $\mathbb{Z}_q^{(2v+2)\times 1}$. Similarly to the proof of Lemma 81, it is also possible to show that fixing the first $v+j+1$ entries of $\vec{\beta}$ determines the $(v+j+2)$-th row of $\mathbf{M}$, for $j=1,\ldots,c$; and that moreover, fixing the first $v+c+1$ entries of $\vec{\beta}$ also determines the last rows of $\vec{\gamma}$ and of $\mathbf{M}$.

Hence, by Lemma 79, we can conclude that the conditional distribution of $\bar{\gamma}_a$ with respect to $\vec{V}$ and to the first $v+c+1$ entries of $\vec{\beta}$, 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 (7.35), and thus the lemma, follows. $\square$

Lemma 90. *Consider the Generalized Reed-Solomon code:*

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

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 $P(\cdot) \in \mathbb{Z}_q^{<n-v}[x]$. 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 ℓ-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 $\ell \leq v$ (just consider the polynomial $Q(x) \doteq P(x)x^{\ell} \in \mathbb{Z}_q^{<n}[x]$). It follows that

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

2. Observe that a vector of $\mathbb{Z}_q^{n-v}$ can be encoded as the coefficients of a polynomial $P(\cdot) \in \mathbb{Z}_q^{<n-v}[x]$. 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. $\square$

Chapter 8

Chosen-Ciphertext Security for Trace-and-Revoke Schemes

8.1 Introduction

As discussed in Chapter 3, Section 3.3, in the public key setting, the only known Trace and Revoke schemes have been constructed by [67, 80] based on the DDH assumption, and achieve public key and message overhead $O(v)$ (where v denotes an upper bound to the number of revoked users). In fact, these schemes are essentially identical: in the following we will refer to the work of [80], who emphasize more the public key nature of their scheme.

Despite providing a simple and elegant scheme, the work of [80] has several noticeable shortcomings. First, the given (informal) notion of security does not address the peculiar features of the revocation setting. Indeed, to show the "security" of revocation, [80] shows the following two claims: (1) the scheme is semantically secure when no users are revoked; (2) no set of v a-priori fixed users can compute the secret key of another user. Clearly, these properties do not imply the security notion we really care about and which informally states: (3) if the adversary controls some set $\mathcal{R}$ of up to v *revoked* users, then the scheme remains semantically secure. Indeed, the scheme of [80] can be shown to satisfy the needed property (3) only when the set $\mathcal{R}$ is chosen by the adversary *non-adaptively*, and in fact only if it is chosen before the adversary learns the public key. Such weak non-adaptive security is clearly insufficient for realistic usages of a public key revocation scheme, since the distributed nature of a broadcast encryption scheme makes it much more prone to adaptive attacks than regular encryption.

Most importantly, the extended scheme of [80] is proven to be IND-ID-CCA-secure when none of the users is corrupted, but stops being such the moment just a single user is corrupted, even if this user is immediately revoked for the rest of the protocol. Again, this is too weak—the scheme should remain IND-ID-CCA-secure *even after many users have been revoked*. As we will see, achieving this strong type of security is very non-trivial, and requires a much more involved scheme than the one proposed by [80].

8.2 Our Results

In this chapter, based on work first published as [36, 35], we introduce for the first time a precise formalization of an appropriate notion of adaptive security for public-key broadcast encryption schemes, for both the IND-ID-CPA and the IND-ID-CCA setting, which naturally models property (3) mentioned above. We construct the first IND-ID-CCA-secure public key broadcast encryption scheme under the DDH assumption, with no random oracle. Our public key scheme is based on the regular Cramer-Shoup encryption [31, 32], but our extension is non-trivial, as we have to resolve some difficulties inherent to the Broadcast Encryption setting. Our IND-ID-CCA-secure scheme requires a constant user storage and a public key size proportional to the revocation threshold v. The length of each ciphertext, and the time to encrypt and decrypt a message are all proportional to $O(v)$.

As a preliminary step, we show how to modify the IND-ID-CPA-scheme of [80] to achieve a much more appropriate notion of adaptive security, while maintaining essentially the same efficiency in all the parameters (up to a factor of 2).

Of independent interest, we provide another scheme achieving the slightly weaker (but still very strong) notion of *generalized chosen-ciphertext* (IND-ID-gCCA) security [75, 2]. As argued in [2], the IND-ID-gCCA security is much more robust to syntactic changes, while still sufficient for all known uses of IND-ID-CCA security. Interestingly, all the examples separating IND-ID-CCA- and IND-ID-gCCA-secure encryption were "artificial" in a sense that they made a more complicated scheme from an already existing IND-ID-CCA-secure encryption. Our work shows the first "natural" separation, but for the setting of *broadcast* public key encryption.

A Note on Traitor Tracing. By slightly modifying standard tracing algorithms from previous schemes (*e.g.* [67, 80]), tracing algorithms can be added to our schemes, thus yielding fully functional *Trace and Revoke* schemes. However, in what follows, we will focus only on Broadcast Encryption (*i.e.* revocation), which is also the main novelty part of our result.

8.3 Constructing Secure Revocation Schemes

In this section, we present three broadcast encryption schemes, achieving IND-ID-CPA, IND-ID-gCCA and IND-ID-CCA security respectively (*cf.* Chapter 4, Section 4.3). Subsequent schemes build on the previous one, in a incremental way, so that it is possible to obtain increasing security at the cost of a slight efficiency loss.

Our proofs follow the structural approach advocated in [32] defining a sequence of attack games $\mathbf{G}_0$, $\mathbf{G}_1$, $\dots$, all operating over the same underlying probability space. Starting from the actual adversarial game $\mathbf{G}_0$, we incrementally make slight modifications to the behavior of the oracles, thus changing the way the adversary's view is computed, while maintaining the view's distributions indistinguishable among the games. While this structural approach takes more space to write down, it is much less error-prone and much more understandable than a slicker "direct argument" (*e.g.*, compare [31, 32]).

8.3.1 IND-ID-CPA Security

As a warm-up, before addressing the more challenging case of chosen ciphertext security, we describe a simpler IND-ID-CPA-secure scheme. Our scheme naturally builds upon previous works [67, 80], but achieves a much more appropriate notion of *adaptive* security, which those previous schemes do not enjoy.

The Key Generation Algorithm. The first step in the key generation algorithm $\mathsf{Setup}(1^\lambda, 1^v)$ is to define a group $\mathbb{G}$ of order q, for a random λ-bit-long prime q such that $p = 2q + 1$ is also prime, in which the DDH assumption is believed to hold. This is accomplished selecting a random prime q with the above two properties and a random element g_1 of order q modulo p: the group $\mathbb{G}$ is then set to be the subgroup of $\mathbb{Z}_p^*$ generated by g_1, *i.e.*

$$\mathbb{G} \doteq \{g_1^i \bmod p : i \in \mathbb{Z}_q\} \subset \mathbb{Z}_p^*$$

A random $w \overset{R}{\leftarrow} \mathbb{Z}_q$ is then chosen and used to compute $g_2 \doteq g_1^w$. (In what follows, all computations are mod q in the exponent, and mod p elsewhere.) Then, the key generation algorithm selects two random v-degree polynomials[1] $Z_1(\cdot)$ and $Z_2(\cdot)$ over $\mathbb{Z}_q$, and computes the values $h_0 \doteq g_1^{Z_{1,0}} \cdot g_2^{Z_{2,0}}, \ldots, h_v \doteq g_1^{Z_{1,v}} \cdot g_2^{Z_{2,v}}$. Finally, the pair $(\mathsf{params_{BE}}, \mathsf{master_{BE}})$ is given in output, where

$$\mathsf{params_{BE}} \doteq \langle g_1, g_2, h_0, \ldots, h_z \rangle$$

$$\mathsf{master_{BE}} \doteq \langle Z_1(\cdot), Z_2(\cdot) \rangle$$

The Registration Algorithm. Each time a new user $i > v$ (in all our schemes, we reserve the first indices $0, \ldots, v$ for "special purposes"), decides to subscribe to the system, the Center provides the user with a decoder box containing the secret key $\mathsf{SK}_i \doteq \langle i, Z_{1,i}, Z_{2,i} \rangle$.

The Encryption Algorithm. The encryption algorithm $\mathsf{Encrypt}$ is given in Figure 8.1. It receives as input the public key $\mathsf{params_{BE}}$, a session key s and a set $\mathcal{R} = \{j_1, \ldots, j_v\}$ of revoked users and returns the enabling block $\mathcal{B}$. If there are less than v revoked users,

[1]For conciseness, we will use the following notation: $Z_{1,i} \doteq Z_1(u)$ and $Z_{2,i} \doteq Z_2(i)$.

the remaining indices are set to $1, \ldots, (v - |\mathcal{R}|)$, which are never given to any "real" user.

```
E1.  $r_1 \overset{R}{\leftarrow} \mathbb{Z}_q$
E2.  $u_1 \overset{R}{\leftarrow} g_1^{r_1}$
E3.  $u_2 \overset{R}{\leftarrow} g_2^{r_1}$
E4.  $H_\ell \overset{R}{\leftarrow} h_\ell^{r_1}, \quad \ell = 0, \ldots, v$
E5.  $H_{j_\ell} \leftarrow \mathsf{EXP\text{-}LI}(0, \ldots, v; H_0, \ldots, H_v)(j_\ell), \quad \ell = 1, \ldots, v$
E6.  $S \leftarrow s \cdot H_0$
E7.  $\mathcal{B} \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}) \rangle$
```

Figure 8.1: Algorithm $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{BE}}, \mathcal{R}, s)$ for the IND-ID-CPA scheme.

The Decryption Algorithm. If a legitimate user i wants to recover the session key embedded in the enabling block $\mathcal{B} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}) \rangle$, he can proceed as in Figure 8.2. If i is a revoked user (*i.e.* $i \in \{j_1, \ldots, j_v\}$), the algorithm fails in step $D2$, since the interpolation points $j_1, \ldots, j_v, i$ are not pairwise distinct.

```
D1.  $H_i \leftarrow u_1^{Z_{1,i}} \cdot u_2^{Z_{2,i}}$
D2.  $s \leftarrow S / \mathsf{EXP\text{-}LI}(j_1, \ldots, j_v, i; H_{j_1}, \ldots, H_{j_v}, H_i)(0)$
```

Figure 8.2: Algorithm $\mathsf{Decrypt}(\mathsf{params}_{\mathsf{BE}}, i, \mathsf{SK}_i, \mathcal{B})$ for the IND-ID-CPA scheme.

Security. As shown in the Theorem 91, the IND-ID-CPA-security of the above scheme relies on the decisional Diffie-Hellman (DDH) assumption.

Theorem 91. *If the* DDH *problem is hard in* $\mathbb{G}$*, then the above broadcast encryption scheme is IND-ID-CPA-secure. In particular, for all probabilistic polynomial-time algorithm* $\mathcal{A}$*, we have that* $IND - ID - CPA.\mathsf{Adv}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \nu(\lambda)$.

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

Game $\mathbf{G}_0$. This game is exactly as the IND-ID-CPA game defined in Chapter 4, Definition 29. Let T_0 be the event that $b = b^*$ in game $\mathbf{G}_0$.

Game $\mathbf{G}_1$. Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that, in game $\mathbf{G}_1$, step $E4$ of the encryption algorithm in Figure 8.1, is replaced with the following:

$$E4'. \quad H_\ell \leftarrow u_1^{Z_{1,\ell}} \cdot u_2^{Z_{2,\ell}}, \quad \ell = 0, \ldots, v.$$

By the properties of the Lagrange Interpolation in the Exponent (*cf.* Chapter 2, Section 2.1.1), it is clear that step $E4'$ computes the same values H_ℓ, $\ell = 0, \ldots, v$ as step $E4$. The point of this change is just to make explicit any functional dependency of the above quantities on u_1 and u_2. Let T_1 be the event that $b = b^*$ in game $\mathbf{G}_1$; clearly, it holds that

$$\Pr[T_0] = \Pr[T_1]. \tag{8.1}$$

Game $\mathbf{G}_2$. To turn game $\mathbf{G}_1$ into game $\mathbf{G}_2$ we make another change to the encryption oracle used in game $\mathbf{G}_1$. In game $\mathbf{G}_2$ steps $E1, E3$ are replaced with the following:

$$E1'. \quad r_1 \xleftarrow{R} \mathbb{Z}_q, \quad r_2 \xleftarrow{R} \mathbb{Z}_q \setminus \{r_1\}$$
$$E3'. \quad u_2 \leftarrow g_2^{r_2}$$

Let T_2 be the event that $b = b^*$ in game $\mathbf{G}_2$. Notice that while in game $\mathbf{G}_1$ the values u_1 and u_2 are obtained using the same value r_1, in game $\mathbf{G}_2$ they are independent subject to $r_1 \neq r_2$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between $\mathbf{G}_1$ and $\mathbf{G}_2$ can be used to construct a probabilistic polynomial-time algorithm $\mathcal{A}_1$ that is able to distinguish Diffie-Hellman tuples from totally random tuples with non negligible advantage. Hence,

$$\big|\Pr[T_2] - \Pr[T_1]\big| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(\lambda). \tag{8.2}$$

Game $\mathbf{G}_3$. To define game $\mathbf{G}_3$, we again modify the encryption oracle as follows:

$$E6'. \quad e \xleftarrow{R} \mathbb{Z}_q, \quad S \leftarrow g_1^e$$

Let T_3 be the event that $b = b^*$ in game $\mathbf{G}_3$. Because of this last change, the challenge no longer contains b, nor does any other information in the adversary's view; therefore, it holds that

$$\Pr[T_3] = \frac{1}{2}. \tag{8.3}$$

Moreover, we can prove (see Lemma 97), that the adversary has the same chances to guess b in both game $\mathbf{G}_2$ and $\mathbf{G}_3$, *i.e.*

$$\Pr[T_3] = \Pr[T_2]. \tag{8.4}$$

Combining Equations (8.1), (8.2), (8.4) and (8.3), adversary $\mathcal{A}$'s advantage can be bounded as:

$$\text{IND-ID-CPA.}\mathsf{Adv}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(\lambda).$$

$\square$

A comparison with the IND-ID-CPA schemes of [67, 80]. Our IND-ID-CPA scheme extends those proposed in [67, 80] by using two generators. This improvement turns out to be crucial: the adaptive security of our IND-ID-CPA scheme hinges heavily upon this change. In particular, the presence of two generators plays a key role in the reduction from the DDH problem mentioned in the description of game $\mathbf{G}_2$ in Theorem 91. More specifically, when setting up the simulation of the IND-ID-CPA game, the two distinct generators used in the public key of the scheme provide the perfect place where to embed the first two elements g_1 and g_2 of the DDH "challenge" at hand. Doing so, the simulator can choose the rest of the public key in a "honest" way, and hence it will know all the corresponding secret keys (*i.e.* the polynomials $Z_1(\cdot)$ and $Z_2(\cdot)$). This in turn allows the simulator to answer any corruption query that the adversary may want to carry out before querying the encryption oracle.

On the contrary, the use of a single generator in both IND-ID-CPA-secure schemes of [67, 80] leads to a reduction in which the simulator does not know the entire secret key (in particular, the constant term of the secret polynomial is unknown to the simulator; *cf.* Theorem 1 of [80]). As a consequence, there seems to be no way to answer corruption queries, so that the IND-ID-CPA game cannot be properly simulated: thus the reduction argument does not go through.

8.3.2 IND-ID-gCCA Security

Once we have constructed an IND-ID-CPA-secure broadcast encryption scheme, it is natural to try to devise an extension achieving IND-ID-CCA security. This was already attempted by [80], but they do not elaborate (neither formally nor informally) on what an "adaptive chosen ciphertext attack" of a broadcast encryption scheme exactly is. As a consequence, in Theorem 3 of [80], the authors only show the security of their scheme against an adversary that does not participate in the system, whereas (as we will argue at the end of this section) their scheme is certainly *not* IND-ID-CCA-secure with respect to even a *single* malicious revoked user.

To achieve IND-ID-CCA security, we will first try to apply the standard technique of [31, 32] to the scheme presented in Chapter 3, Section 8.3.1. Unfortunately, this natural approach does not completely solve the IND-ID-CCA problem; still, it leads us to an interesting scheme that achieves the (sligthly weaker) notion of generalized chosen ciphertext security.

The Key Generation Algorithm. As before, the first task of the key generation algorithm is to select a random group $\mathbb{G} \subset \mathbb{Z}_p^*$ of prime order q and two random generators $g_1, g_2 \in \mathbb{G}$. Then, $\mathsf{Setup}(1^\lambda, 1^v)$ selects six random v-degree polynomials[2] $X_1(\cdot)$, $X_2(\cdot)$, $Y_1(\cdot)$, $Y_2(\cdot)$, $Z_1(\cdot)$ and $Z_2(\cdot)$ over $\mathbb{Z}_q$, and, for $\ell = 0, \ldots, v$, computes the values $c_\ell \doteq g_1^{X_{1,\ell}} \cdot g_2^{X_{2,\ell}}$, $d_\ell \doteq g_1^{Y_{1,\ell}} \cdot g_2^{Y_{2,\ell}}$, and $h_\ell \doteq g_1^{Z_{1,\ell}} \cdot g_2^{Z_{2,\ell}}$ Finally, Setup chooses at random a hash function $\mathcal{H}$ from a family $\mathcal{F}$ of collision resistant hash functions,[3] and outputs the

[2]For conciseness, we will use the following notation: $X_{1,i} \doteq X_1(i), X_{2,i} \doteq X_2(i), Y_{1,i} \doteq Y_1(i), Y_{2,i} \doteq Y_2(i), Z_{1,i} \doteq Z_1(i)$ and $Z_{2,i} \doteq Z_2(i)$.

[3]Recall, it is hard to find $x \neq y$ such that $\mathcal{H}(x) = \mathcal{H}(y)$ for a random member $\mathcal{H}$ of $\mathcal{F}$.

pair $(\mathsf{params}_{\mathsf{BE}}, \mathsf{master}_{\mathsf{BE}})$, where

$$\mathsf{params}_{\mathsf{BE}} \doteq \langle g_1, g_2, c_0, \ldots, c_v, d_0, \ldots, d_v, h_0, \ldots, h_v, \mathcal{H} \rangle,$$
$$\mathsf{master}_{\mathsf{BE}} \doteq \langle X_1, X_2, Y_1, Y_2, Z_1, Z_2 \rangle.$$

The Registration Algorithm. Each time a new user $i > v$ subscribes to the system, the Center provides the user with a decoder box containing the secret key $\mathsf{SK}_i \doteq \langle i, X_{1,i}, X_{2,i}, Y_{1,i}, Y_{2,i}, Z_{1,i}, Z_{2,i} \rangle$.

The Encryption Algorithm. Using the idea of [31, 32], in order to obtain non-malleable ciphertexts, we "tag" each encrypted message so that it can be verified before proceeding with the actual decryption. In the broadcast encryption scenario, where each user has a different decryption key, the tag cannot be a single point — we need to distribute an entire EXP-polynomial $\mathcal{V}(x)$. This is accomplished appending $v+1$ tags to the ciphertext: each user i first computes the tag v_i using his private key and then verifies the validity of the ciphertext by checking the interpolation of the $v+1$ values in point i against its v_i.

The encryption algorithm $\mathsf{Encrypt}$ receives as input the public key $\mathsf{params}_{\mathsf{BE}}$, the session key s to be embedded within the enabling block and a set $\mathcal{R} = \{j_1, \ldots, j_v\}$ of revoked users. It proceeds as described in Figure 8.3, and finally it outputs $\mathcal{B}$.

The Decryption Algorithm. If a legitimate user i wants to recover the session key embedded in the enabling block $\mathcal{B} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}), t_0, \ldots, t_v \rangle$, he can proceed as in Figure 8.4. If i is a revoked user, the algorithm fails in step $D6$, since the interpolation points $j_1, \ldots, j_v, i$ are not pairwise distinct.

Security. As mentioned above, the presence of many decryption keys leads to the use of an EXP-polynomial $\mathcal{V}(x)$ to tag the encryption of the message. This in turn makes the ciphertext malleable: since each user i can verify the value of $\mathcal{V}(x)$ only in one point, the adversary can modify the t_j's values and construct a different EXP-polynomial $\mathcal{V}'(x)$ intersecting $\mathcal{V}(x)$ at point i—thus fooling user i to accept as valid a corrupted ciphertext. In the next section we show a non-trivial solution to this problem; here, we assess the IND-ID-gCCA-security of the broadcast encryption scheme presented above. As already

```
E1.  $r_1 \xleftarrow{R} \mathbb{Z}_q$
E2.  $u_1 \leftarrow g_1^{r_1}$
E3.  $u_2 \leftarrow g_2^{r_1}$
E4.  $H_\ell \leftarrow h_\ell^{r_1}, \quad \ell = 0, \ldots, v$
E5.  $H_{j_\ell} \leftarrow \mathsf{EXP\text{-}LI}(0, \ldots, v; H_0, \ldots, H_v)(j_\ell), \quad \ell = 1, \ldots, v$
E6.  $S \leftarrow s \cdot H_0$
E7.  $\alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}))$
E8.  $t_\ell \leftarrow c_\ell^{r_1} \cdot d_\ell^{r_1 \alpha}, \quad \ell = 0, \ldots, v$
E9.  $\mathcal{B} \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}), t_0, \ldots, t_v \rangle$
```

Figure 8.3: Algorithm $\mathsf{Encrypt}(\mathsf{params}_{\mathsf{BE}}, \mathcal{R}, s)$ for the IND-ID-gCCA scheme.

```
D1.  $\alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}))$
D2.  $\bar{t}_i \leftarrow u_1^{X_{1,i} + Y_{1,i}\alpha} \cdot u_2^{X_{2,i} + Y_{2,i}\alpha}$
D3.  $t_i \leftarrow \mathsf{EXP\text{-}LI}(0, \ldots, v; t_0, \ldots, t_v)(i)$
D4.  **if** $t_i = \bar{t}_i$
D5.  **then** $H_i \leftarrow u_1^{Z_{1,i}} \cdot u_2^{Z_{2,i}}$
D6.           $s \leftarrow S / \mathsf{EXP\text{-}LI}(j_1, \ldots, j_v, i; H_{j_1}, \ldots, H_{j_v}, H_i)(0)$
D7.           **return** $s$
D8.  **else** **return** $\perp$
```

Figure 8.4: Algorithm $\mathsf{Decrypt}(\mathsf{params}_{\mathsf{BE}}, i, \mathsf{SK}_i, \mathcal{B})$ for the IND-ID-gCCA scheme.

discussed in Chapter 4, to this aim it is necessary to introduce a family of equivalence relations $\{\Re_i(\cdot, \cdot)\}$: intuitively, two ciphertexts $\mathcal{B}$ and $\mathcal{B}'$ are equivalent for user i if they have the same "data" components, and the tag "relevant to user i" is correctly verified, *i.e.* $t_i = t'_i$ (even though other "irrelevant" tags could be different). Clearly, this relation is efficiently computable and i-decryption-respecting.

Definition 92 (Equivalence Relation)**.** *Consider the* EXP*-polynomials*

$$\mathcal{V}(x) \doteq \mathsf{EXP\text{-}LI}(0,\ldots,v;t_0,\ldots,t_v)(x),$$
$$\mathcal{V}'(x) \doteq \mathsf{EXP\text{-}LI}(0,\ldots,v;t'_0,\ldots,t'_v)(x).$$

Given a user i*, and the two enabling blocks*

$$\mathcal{B} \doteq \langle S,u_1,u_2,(j_1,H_{j_1}),\ldots,(j_v,H_{j_v}),t_0,\ldots,t_v\rangle$$
$$\mathcal{B}' \doteq \langle S,u_1,u_2,(j_1,H_{j_1}),\ldots,(j_v,H_{j_v}),t'_0,\ldots,t'_v\rangle$$

we say that $\mathcal{B}$ *is* equivalent to $\mathcal{B}'$ with respect to user i*, and we write* $\Re_i(\mathcal{B},\mathcal{B}')$*, if the two* EXP*-polynomials* $\mathcal{V}(x)$ *and* $\mathcal{V}'(x)$ *intersect at point* i*,* i.e. $t_i = \mathcal{V}(i) = \mathcal{V}'(i) = t'_i$.

Theorem 93. *If the* DDH *Problem is hard in* $\mathbb{G}$ *and* $\mathcal{H}$ *is chosen from a collision-resistant hash functions family* $\mathcal{F}$*, then the above broadcast encryption scheme is IND-ID-gCCA-secure, under the family of equivalence relations* $\{\Re_i(\cdot,\cdot)\}$ *defined in Definition 92.*

Proof. To prove this theorem, we pursue the same approach as in the proof of Theorem 91, where the starting scenario of the sequence of games is defined as in the definition of the adaptive IND-ID-gCCA attack.

Game $\mathbf{G}_0$. This game is exactly as the IND-ID-gCCA game defined in Chapter 4. Let T_0 be the event that $b = b^*$ in game $\mathbf{G}_0$.

Game $\mathbf{G}_1$. Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that, in game $\mathbf{G}_1$, steps $E4, E8$ of the encryption algorithm in Figure 8.3, are replaced with the following:

$$E4'.\quad H_\ell \leftarrow u_1^{Z_{1,\ell}} \cdot u_2^{Z_{2,\ell}}, \quad \ell = 0,\ldots,v$$
$$E8'.\quad t_\ell \leftarrow u_1^{X_{1,\ell}+Y_{1,\ell}\alpha} \cdot u_2^{X_{2,\ell}+Y_{2,\ell}\alpha} \quad \ell = 0,\ldots,v$$

By the properties of the Lagrange Interpolation in the Exponent, it is clear that step $E4'$ computes the same values H_{j_ℓ}, $\ell = 0,\ldots,v$ as steps $E4$; similarly, step $E8'$ computes the same values t_ℓ, $\ell = 0,\ldots,v$ as step $E8$. The point of these changes is just to make explicit any functional dependency of the above quantities on u_1 and u_2.

Let T_1 be the event that $b = b^*$ in game $\mathbf{G}_1$. Clearly, it holds that

$$\Pr[T_0] = \Pr[T_1]. \tag{8.5}$$

Game $\mathbf{G}_2$. To turn game $\mathbf{G}_1$ into game $\mathbf{G}_2$ we make another change to the encryption oracle used in game $\mathbf{G}_1$. In game $\mathbf{G}_2$ steps $E1, E3$ are replaced with the following:

$$E1'. \quad r_1 \xleftarrow{R} \mathbb{Z}_q, \quad r_2 \xleftarrow{R} \mathbb{Z}_q \setminus \{r_1\}$$

$$E3'. \quad u_2 \leftarrow g_2^{r_2}$$

Let T_2 be the event that $b = b^*$ in game $\mathbf{G}_2$. Notice that while in game $\mathbf{G}_1$ the values u_1 and u_2 are obtained using the same value r_1, in game $\mathbf{G}_2$ they are independent subject to $r_1 \neq r_2$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between $\mathbf{G}_1$ and $\mathbf{G}_2$ can be used to construct a probabilistic polynomial-time algorithm $\mathcal{A}_1$ that is able to distinguish Diffie-Hellman tuples from totally random tuples with non negligible advantage. Hence,

$$\big| \Pr[T_2] - \Pr[T_1] \big| \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(\lambda). \tag{8.6}$$

Game $\mathbf{G}_3$. To define game $\mathbf{G}_3$ we slightly modify the decryption oracle: instead of using the algorithm in Figure 8.4, in game $\mathbf{G}_3$ steps $D2, D4, D5$ are replaced with the following:

$$D2'. \quad \bar{t}_i \leftarrow u_1^{(X_{1,i}+Y_{1,i}\alpha)+(X_{2,i}+Y_{2,i}\alpha)\cdot w}$$

$$D4'. \quad \textbf{if } (u_2 = u_1^w \wedge t_i = \bar{t}_i)$$

$$D5'. \quad \textbf{then } H_i \leftarrow u_1^{Z_{1,i}+Z_{1,i}\cdot w}$$

The rationale behind these changes is that we want to strengthen the condition that the enabling block has to meet in order to be considered valid and hence to be decrypted. This will make it easier to show the security of the scheme; however, for these changes to be useful, there should be no observable difference in the way invalid enabling blocks are "caught" in games $\mathbf{G}_2$ and $\mathbf{G}_3$. To make it formal, we now introduce the following two events: let T_3 be the event that $b = b^*$ in game $\mathbf{G}_3$, and let R_3 be the event that

$\mathcal{A}$ submits some decryption query that would have been decrypted in game $\mathbf{G}_2$ but is rejected in game $\mathbf{G}_3$; in other words, R_3 is the event that some decryption query that would have passed the test in step $D4$ of the decryption oracle used in game $\mathbf{G}_2$, fails to pass the test in step $D4'$ used in game $\mathbf{G}_3$. Clearly, $\mathbf{G}_2$ and $\mathbf{G}_3$ are identical until event R_3 occurs; hence, if R_3 never occurs, the adversary has the same chances to win in both the two games, *i.e.* using Lemma 95),

$$T_3 \wedge \neg R_3 \equiv T_2 \wedge \neg R_3 \Rightarrow \left| \Pr[T_3] - \Pr[T_2] \right| \leq \Pr[R_3]. \tag{8.7}$$

To bound the last probability, we consider two more games, $\mathbf{G}_4$ and $\mathbf{G}_5$.

Game $\mathbf{G}_4$. To define game $\mathbf{G}_4$, we again modify the encryption oracle as follows:

$$E6'. \quad e \overset{R}{\leftarrow} \mathbb{Z}_q, \quad S \leftarrow g_1^e$$

Let T_4 be the event that $b = b^*$ in game $\mathbf{G}_4$. Because of this last change, the challenge no longer contains the bit b, nor does any other information in the adversary's view; therefore, it holds that

$$\Pr[T_4] = \frac{1}{2}. \tag{8.8}$$

Let R_4 be the event that $\mathcal{A}$ submits some decryption query that would have been decrypted in game $\mathbf{G}_2$ but is rejected in game $\mathbf{G}_4$; in other words, R_4 is the event that some decryption query that would have passed the test in step $D4$ of the decryption oracle used in game $\mathbf{G}_2$, fails to pass the test in step $D4'$ used in game $\mathbf{G}_4$. In Lemma 98, we show that those events happen with the same probability as the corresponding events of game $\mathbf{G}_3$, *i.e.*

$$\Pr[T_4] = \Pr[T_3] \tag{8.9}$$

$$\Pr[R_4] = \Pr[R_3]. \tag{8.10}$$

Game $\mathbf{G}_5$. In this game, we again modify the decryption algorithm, adding the following *special rejection rule*, whose goal is to prevent the adversary from submitting illegal enabling blocks to the decryption oracle, once she has received her challenge.

After $\mathcal{A}$ receives her challenge

$$\mathcal{B}^* = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_v^*, H_{j_v^*}), t_0^*, \ldots, t_v^* \rangle$$

the decryption oracle rejects any query $\langle i, \mathcal{B} \rangle$, with

$$\mathcal{B} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}), t_0, \ldots, t_v \rangle$$

such that

$$\langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}) \rangle \neq \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_v^*, H_{j_v^*}) \rangle$$

but $\alpha = \alpha^*$, and it does so before executing the test in step $D4'$.

Notice that in the IND-ID-gCCA setting the adversary is not allowed to query the decryption oracle $\mathsf{Decrypt}(\mathsf{params}_{\mathsf{BE}}, i, \mathsf{SK}_i, \mathcal{B})$ on enabling blocks $\Re_i$-equivalent to the challenge $\mathcal{B}^*$. Therefore, when the *special rejection rule* is applied, we already know that it holds $\neg\Re_i(\mathcal{B}, \mathcal{B}^*)$.

Let C_5 be the event that the adversary submits a decryption query that is rejected using the above *special rejection rule*; let R_5 be the event that $\mathcal{A}$ submits some decryption query that would have passed the test in step $D4$ of the decryption oracle used in game $\mathbf{G}_2$, but fails to pass the test in step $D4'$ used in game $\mathbf{G}_5$. Notice that this implies that such a query passed the $\Re_i$-equivalence test and the *special rejection rule*, because otherwise step $D4'$ wouldn't have been executed. Clearly, $\mathbf{G}_4$ and $\mathbf{G}_5$ are identical until event C_5 occurs, *i.e.*

$$R_5 \wedge \neg C_5 \equiv R_4 \wedge \neg C_5 \Rightarrow \left| \Pr[R_5] - \Pr[R_4] \right| \leq \Pr[C_5] \tag{8.11}$$

where the implication follows from Lemma 95.

Our final task is to show that events C_5 and R_5 occur with negligible probability: while the argument to bound event C_5 is based on the collision resistance assumption for the family $\mathcal{F}$ (using a standard reduction argument, we can construct a probabilistic polynomial-time algorithm $\mathcal{A}_2$ that breaks the collision resistance assumption with non negligible advantage), the argument to bound event R_5 hinges upon the fact that the adversary is not allowed to submit queries that are "$\Re_i$-related" to her challenge, and upon information-theoretic considerations (as proven in Lemma 99). From these considerations, we obtain that

$$\Pr[C_5] \leq \varepsilon_{CR} \tag{8.12}$$

$$\Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q} \tag{8.13}$$

where ε_{CR} is a negligible quantity and $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary.

Combining Equations (8.5), (8.6), (8.7), (8.8), (8.9), (8.10), (8.11), (8.12) and (8.13), adversary $\mathcal{A}$'s advantage can be bounded as:

$$\text{IND-ID-gCCA}.\mathsf{Adv}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(\lambda) + \varepsilon_{CR} + Q_{\mathcal{A}}(\lambda)/q.$$

□

A comparison with the IND-ID-CCA attempt of [80]. As already noticed when describing the encryption algorithm, in the broadcast encryption setting it is not safe to use a single point as validating tag; it is for that reason that in our IND-ID-gCCA solution we use a validating EXP-polynomial $\mathcal{V}(x)$. On the contrary, in [80] the authors didn't recognize the inherent insecurity of using a single tag and proposed a scheme that not only uses a single generator, but also tags ciphertexts with just one point. Consequently, the information distributed to the users of the system to enable them to check the validity of a given ciphertext (namely, x_1, x_2, y_1, y_2 in the notation of [80]) is the *same* for all participants. To verify the validity of a ciphertext $\mathcal{B}$, a user recomputes the tag $\bar{t} = F_a^{x_1+y_1\alpha} \cdot F_b^{x_2+y_2\alpha}$ from quantities present in the ciphertext itself and from the secret information x_1, x_2, y_1, y_2 (common to all users). The value $\bar{t}$ is then compared against the tag t in $\mathcal{B}$.

This means that revoking a user does not affect his/her ability to check the validity of a ciphertext; furthermore, validating a ciphertext is effectively equivalent to computing the corresponding tag. This implies that any revoked user is able to construct new, legal ciphertexts from any encrypted message; in other words, ciphertexts are *malleable* and hence the scheme cannot be IND-ID-CCA secure. More precisely, even an adversary that non-adaptively corrupts just a single user, can break the scheme with a single decryption query: upon receiving the challenge ciphertext from the encryption oracle, the adversary changes it in some easily-reversible way, computes the proper tag (exploiting the knowledge of x_1, x_2, y_1, y_2 that she got from the revoked user) and asks the decryption oracle to decrypt such modified ciphertext. Once the adversary gets the decrypted message back, she can easily tell which message was hidden within her challenge, breaking the scheme.

8.3.3 IND-ID-CCA Security

In Section 8.3.2, we saw how a direct application of the standard technique of [31, 32] does not provide a complete solution to the IND-ID-CCA problem, but only suffices for IND-ID-gCCA security. As proven in Lemma 99, the restriction imposed by the IND-ID-gCCA attack (namely, forbidding the adversary to submit decryption queries $\langle i, \mathcal{B} \rangle$ such that $\Re_i(\mathcal{B}, \mathcal{B}^*)$ holds) is essential for the security of the previous broadcast encryption scheme. Indeed, given a challenge $\mathcal{B}^*$ with tag sequence $t_0, \ldots, t_v$, it is trivial to come up with a different sequence $t'_0, \ldots, t'_v$ such that $t_i = t'_i$, resulting in a "different" enabling block $\mathcal{B}' \neq \mathcal{B}^*$: however, $\mathsf{Decrypt}(i, \mathcal{B}^*) = \mathsf{Decrypt}(i, \mathcal{B}')$, allowing the adversary to "break" the IND-ID-CCA security.

Although we feel that IND-ID-gCCA security is enough for most applications of broadcast encryption schemes, it is possible to extend the broadcast encryption scheme presented in Section 8.3.2 to obtain IND-ID-CCA security (with only a slight efficiency loss). The modified scheme, presented in this section, maintains the same Setup and Register algorithms described before; the essential modifications involve the operations used to construct the enabling block. In particular, to achieve IND-ID-CCA security, it is necessary to come up with some trick to make the tag sequence $t_0, \ldots, t_v$ non-malleable.

To this aim, we will use any secure (deterministic) *message authentication code* (MAC) [76] to guarantee the integrity of the entire sequence. In fact, we only need any *one-time* MAC, satisfying the following simple property: given a (unique) correct value $\mathsf{MAC}_k(m)$ for some message m (under key k), it is infeasible to come up with a correct (unique) value of $\mathsf{MAC}_k(m')$, for any $m' \neq m$.

The Encryption Algorithm. The encryption algorithm $\mathsf{Encrypt}$ receives as input the public key $\mathsf{params_{BE}}$, the session key s to be embedded within the enabling block and a set $\mathcal{R} = \{j_1, \ldots, j_v\}$ of revoked users. To construct the enabling block $\mathcal{B}$, the encryption algorithm (defined in Figure 8.3) operates similarly to the IND-ID-gCCA encryption algorithm: the main difference is that now a MAC key k, randomly chosen from the MAC key space $\mathcal{K}$, is used to MAC the tag sequence $t_0, \ldots, t_v$, and is encapsulated within $\mathcal{B}$ along with the session key s.

```
E1.  $r_1 \overset{R}{\leftarrow} \mathbb{Z}_q$
E2.  $u_1 \leftarrow g_1^{r_1}$
E3.  $u_2 \leftarrow g_2^{r_1}$
E4.  $H_\ell \leftarrow h_\ell^{r_1}, \quad \ell = 0, \ldots, v$
E5.  $H_{j_\ell} \leftarrow \mathsf{EXP\text{-}LI}(0, \ldots, v; H_0, \ldots, H_v)(j_\ell), \quad \ell = 1, \ldots, v$
E6.  $k \overset{R}{\leftarrow} \mathcal{K}$
E7.  $S \leftarrow (s \parallel k) \cdot H_0$
E8.  $\alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}))$
E9.  $t_\ell \leftarrow c_\ell^{r_1} \cdot d_\ell^{r_1 \alpha}, \quad \ell = 0, \ldots, v$
E10. $\tau \leftarrow \mathsf{MAC}_k(t_0, \ldots, t_v)$
E11. $\mathcal{B} \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}), t_0, \ldots, t_v, \tau \rangle$
```

Figure 8.5: Algorithm $\mathsf{Encrypt}(\mathsf{params_{BE}}, \mathcal{R}, s)$ for the IND-ID-CCA scheme.

The Decryption Algorithm. If a legitimate user i wants to recover the session key embedded in the enabling block $\mathcal{B} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}) \rangle$, he can proceed as in Figure 8.6. If i is a revoked user, the algorithm fails in step $D6$, since the interpolation points $j_1, \ldots, j_v, i$ are not pairwise distinct.

```
D1.  $\alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}))$
D2.  $\bar{t}_i \leftarrow u_1^{X_{1,i}+Y_{1,i}\alpha} \cdot u_2^{X_{2,i}+Y_{2,i}\alpha}$
D3.  $t_i \leftarrow \mathsf{EXP\text{-}LI}(0, \ldots, v; t_0, \ldots, t_v)(i)$
D4.  if $t_i = \bar{t}_i$
D5.  then $H_i \leftarrow u_1^{Z_{1,i}} \cdot u_2^{Z_{2,i}}$
D6.       $s \parallel k \leftarrow S / \mathsf{EXP\text{-}LI}(j_1, \ldots, j_v, i; H_{j_1}, \ldots, H_{j_v}, H_i)(0)$
D7.       extract $s$ and $k$ from $s \parallel k$
D8.       if $\tau \neq \mathsf{MAC}_k(t_0, \ldots, t_v)$
D9.       then return $\perp$
D10.      else return $s$
D11. else return $\perp$
```

Figure 8.6: Algorithm $\mathsf{Decrypt}(\mathsf{params}_{\mathsf{BE}}, i, \mathsf{SK}_i, \mathcal{B})$ for the IND-ID-CCA scheme.

Security. The security analysis for this scheme is very subtle, because there is the risk of circularity in the use of the MAC key k. Namely, k is part of the ciphertext (since it is encapsulated, along with the session key s, within S); this means that α, the hash of the ciphertext, depends on k (at least information-theoretically), and thus the sequence of tags depends on k. In other words, we are MAC-ing something that depends on the MAC key k, which could be a problem. Luckily, the information-theoretic nature of the structural approach to the security analysis that we are pursuing (following [32]) allows us to prove that actually k is completely hidden within S, so that MAC-ing the resulting tag with k is still secure.

The solution to the IND-ID-CCA problem for broadcast encryption schemes and the relative security analysis can be viewed as the main technical contribution of this paper; at the same time, the capability to resolve the apparent circularity in the use of the

MAC demonstrates the importance of providing a formal model and precise definitions, without which it would have been much harder to devise a correct proof of security for the above scheme.

Theorem 94. *If the* DDH *Problem is hard in* $\mathbb{G}$*,* $\mathcal{H}$ *is chosen from a collision-resistant hash functions family* $\mathcal{F}$ *and* MAC *is a one-time message authentication code, then the above broadcast encryption scheme is IND-ID-CCA-secure.*

Proof. The proof proceeds defining a sequence of games similar to that presented in Theorem 93. The definition of games $\mathbf{G}_0, \ldots, \mathbf{G}_5$ closely follow the exposition given in Theorem 93: however, the statements of all lemmas (and their proofs) need to be changed to accommodate for the use of the MAC. In particular, we can easily state and prove a lemma analogous to Lemma 98, where the only difference is the presence of information about the MAC key k in the challenge (see Lemma 102). More importantly, to bound the probability $\Pr[R_5]$ we introduce a new game $\mathbf{G}_6$ to deal with the use of the MAC in the enabling block, while a lemma similar to Lemma 99 is used to bound the probability of event R_6 defined in game $\mathbf{G}_6$.

Game $\mathbf{G}_6$. To define this game, we modify the decryption algorithm, adding the following *second special rejection rule*, whose goal is to detect illegal enabling blocks submitted by the adversary to the decryption oracle, once she has received her challenge. Notice that, while the *special rejection rule*, defined in game $\mathbf{G}_5$, is used to reject adversary's queries aiming at exploiting any weakness in the collision-resistant hash family $\mathcal{F}$, the *second special rejection rule* is used to reject ciphertexts aiming at exploiting any weakness in the MAC scheme.

After $\mathcal{A}$ receives her challenge

$$\mathcal{B}^* = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_v^*, H_{j_v^*}), t_0^*, \ldots, t_v^*, \tau^* \rangle$$

the decryption oracle rejects any query $\langle i, \mathcal{B} \rangle$, with

$$\mathcal{B} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}), t_0, \ldots, t_v, \tau \rangle$$

such that

$$\langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}) \rangle = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_v^*, H_{j_v^*}) \rangle$$

and $(t_0, \ldots, t_v) \neq (t_0^*, \ldots, t_v^*)$, but $\tau = \mathsf{MAC}_{k^*}(t_0, \ldots, t_v)$, and it does so before executing the test in step $D4'$, and before applying the *special rejection rule*.

Let M_6 be the event that the adversary submits a decryption query that is rejected in game $\mathbf{G}_6$ using the *second special rejection rule*; let C_6 be the event that the adversary submits a decryption query that is rejected in game $\mathbf{G}_6$ using the *special rejection rule*; let R_6 be the event that $\mathcal{A}$ submits some decryption query that would have passed both the test in step $D4$ and in step $D8$ of the decryption oracle used in game $\mathbf{G}_2$, but fails to pass the test in step $D4'$ used in game $\mathbf{G}_6$. Notice that this implies that such a query passed both the *second special rejection rule* and the *special rejection rule*, because otherwise step $D4'$ wouldn't have been executed at all.

Event M_6 is closely related to the security of the one time MAC used in the scheme; in particular, any difference in behavior between game $\mathbf{G}_5$ and game $\mathbf{G}_6$ can be used to construct a probabilistic polynomial-time algorithm $\mathcal{A}_3$ that is able to forge a legal authentication code under a one-message attack with non-negligible probability, thus breaking the MAC scheme. Hence, for some negligible $\varepsilon_{\mathsf{MAC}}$,

$$\Pr[M_6] \leq \varepsilon_{\mathsf{MAC}}. \tag{8.14}$$

Moreover, since $\mathbf{G}_5$ and $\mathbf{G}_6$ are identical until event M_6 occurs, if it doesn't occur at all, they will proceed identically; *i.e.*, by Lemma 95:

$$C_6 \wedge \neg M_6 \equiv C_5 \wedge \neg M_6 \Rightarrow \big|\Pr[C_6] - \Pr[C_5]\big| \leq \Pr[M_6], \tag{8.15}$$

$$R_6 \wedge \neg M_6 \equiv R_5 \wedge \neg M_6 \Rightarrow \big|\Pr[R_6] - \Pr[R_5]\big| \leq \Pr[M_6]. \tag{8.16}$$

Our final task is to bound the probability that events C_6 and R_6 occur: the argument to bound $\Pr[C_6]$ is based on the collision resistance assumption for the family $\mathcal{F}$, while the argument to bound $\Pr[R_6]$ hinges upon information-theoretic considerations (as proven in Lemma 103). From those facts, we obtain that

$$\Pr[C_6] \leq \varepsilon_{CR} \tag{8.17}$$

$$\Pr[R_6] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q} \tag{8.18}$$

where ε_{CR} is a negligible quantity and $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary.

Combining Equations (8.5), (8.6), (8.7), (8.8), (8.9), (8.10), (8.11), (8.12), (8.13), (8.14), (8.15), (8.16), (8.17) and (8.18), adversary $\mathcal{A}$'s advantage can be bounded as:

$$IND-ID-CCA.\mathsf{Adv}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \mathsf{AdvDDH}_{\mathbb{G},\mathcal{A}}(\lambda) + \varepsilon_{CR} + 2\varepsilon_{\mathsf{MAC}} + Q_{\mathcal{A}}(\lambda)/q.$$

□

8.4 Proofs of the Technical Lemmas

The proofs of the following lemmas is based on the same techniques used in [32]; the main tools are the following technical lemmas:

Lemma 95. *If U_1, U_2 and F are events such that $(U_1 \wedge \neg F)$ and $(U_2 \wedge \neg F)$ are equivalent events, then*

$$|\Pr[U_1] - \Pr[U_2]| \leq \Pr[F].$$

Lemma 96. *Let k,n be integers with $1 \leq k \leq n$, and let K be a finite field. Consider a probability space with random variables $\vec{\alpha} \in K^{n\times 1}, \vec{\beta} = (\beta_1, \ldots, \beta_k)^T \in K^{k\times 1}, \vec{\gamma} \in K^{k\times 1}$, and $\boldsymbol{M} \in K^{k\times n}$, such that $\vec{\alpha}$ is uniformly distributed over $K^n, \vec{\beta} = \boldsymbol{M}\vec{\alpha} + \vec{\gamma}$, and for $1 \leq i \leq k$, the first i-th rows of $\boldsymbol{M}$ and $\vec{\gamma}$ are determined by $\beta_1, \ldots, \beta_{i-1}$. Then, conditioning on any fixed values of $\beta_1, \ldots, \beta_{k-1}$ such that the resulting matrix $\boldsymbol{M}$ has rank k, the value of β_k is uniformly distributed over K in the resulting conditional probability space.*

In what follows, we will denote with Coins the coin tosses of $\mathcal{A}$ and we define

$$\mathbf{X}_\ell \doteq X_{1,\ell} + wX_{2,\ell}, \quad \mathbf{Y}_\ell \doteq Y_{1,\ell} + wY_{2,\ell}, \quad \mathbf{Z}_\ell \doteq Z_{1,\ell} + wZ_{2,\ell}, \qquad \ell = 0, \ldots, v$$

8.4.1 Proof of the Lemma for IND-ID-CPA Security

Lemma 97. $\Pr[T_4] = \Pr[T_3]$

Proof. Consider the quantity $\vec{V} := (\mathsf{Coins}, w,\ \mathbf{Z}_1,\ \ldots,\ \mathbf{Z}_v,\ b, r_1^*, r_2^*)$ and the value $\mathbf{Z}_0$. According to the specification of games $\mathbf{G}_2$ and $\mathbf{G}_3$, $\vec{V}$ and $\mathbf{Z}_0$ assume the same value in both games. Let us now consider the value $e^* = \log_{g_1} S^*$: unlike the previous two quantities, e^* assumes different values in the above two games. In particular, while in game $\mathbf{G}_2$ e^* contains information about the session key s_b, in game $\mathbf{G}_3$ e^* is just a random value: let us denote with $[e^*]_2$ and $[e^*]_3$ the values of e^* in game $\mathbf{G}_2$ and game $\mathbf{G}_3$, respectively.

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

According to the specification of game $\mathbf{G}_3$, $[e^*]_3$ is chosen uniformly over $\mathbb{Z}_q$, independently from $\vec{V}$ and $\mathbf{Z}_0$. Hence, to reach the thesis, it suffices to prove that the distribution of $[e^*]_2$, conditioned on $\vec{V}$ and $\mathbf{Z}_0$, is also uniform in $\mathbb{Z}_q$.

In game $\mathbf{G}_2$, the quantities $(\vec{V}, \mathbf{Z}_0, [e^*]_2)$ are related according to the following matrix equation:

$$\begin{pmatrix} \mathbf{Z}_0 \\ [e^*]_2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ r_1^* & wr_2^* \end{pmatrix}}_{\mathbf{M}} \cdot \begin{pmatrix} Z_{1,0} \\ Z_{2,0} \end{pmatrix} + \begin{pmatrix} 0 \\ \log_{g_1} s_b \end{pmatrix}$$

where $det(\mathbf{M}) = w(r_2^* - r_1^*) \neq 0$, provided $r_2^* \neq r_1^*$.

As soon as we fix the value of $\vec{V}$, the matrix $\mathbf{M}$ is completely fixed, but the values $Z_{1,0}$ and $Z_{2,0}$ are still uniformly and independently distributed over $\mathbb{Z}_q$. Now, fixing a value for $\mathbf{Z}_0$ also fixes a value for s_b; hence, by Lemma 96, we can conclude that the conditioned distribution of $[e^*]_2$, w.r.t. $\vec{V}$ and $\mathbf{Z}_0$, is also uniform over $\mathbb{Z}_q$. □

8.4.2 Proofs of Lemmas for IND-ID-gCCA Security

Lemma 98. $\Pr[T_4] = \Pr[T_3]$ *and* $\Pr[R_4] = \Pr[R_3]$

Proof. Consider the quantity $\vec{V} := (\mathsf{Coins}, \mathcal{H}, w,\ X_{1,0},\ X_{2,0},\ \ldots,\ X_{1,v},\ X_{2,v},\ Y_{1,0},\ Y_{2,0},\ \ldots,\ Y_{1,v},\ Y_{2,v},\ \mathbf{Z}_1,\ \ldots,\ \mathbf{Z}_v,\ b, r_1^*, r_2^*)$ and the value $\mathbf{Z}_0$. Introducing similar notations as in Lemma 97 and reasoning as above, we can notice that event T_3 solely depends on $(\vec{V}, \mathbf{Z}_0, [e^*]_3)$ and that event T_4 solely depends on $(\vec{V}, \mathbf{Z}_0, [e^*]_4)$. Moreover, event T_3 depends on $(\vec{V}, \mathbf{Z}_0, [e^*]_3)$ according to the same functional dependence of event T_4 upon $(\vec{V}, \mathbf{Z}_0, [e^*]_4)$. The same considerations hold for events R_3 and R_4. Therefore, to prove the lemma, it suffices to show that $(\vec{V}, \mathbf{Z}_0, [e^*]_3)$ and $(\vec{V}, \mathbf{Z}_0, [e^*]_4)$ have the same distribution.

According to the specification of game $\mathbf{G}_4$, $[e^*]_4$ is chosen uniformly over $\mathbb{Z}_q$, independently from $\vec{V}$ and $\mathbf{Z}_0$. Hence, to reach the thesis, it suffices to prove that the distribution of $[e^*]_3$, conditioned on $\vec{V}$ and $\mathbf{Z}_0$, is also uniform in $\mathbb{Z}_q$.

In game $\mathbf{G}_3$, the quantities $(\vec{V}, \mathbf{Z}_0, [e^*]_3)$ are related according to the following matrix equation:

$$\begin{pmatrix} \mathbf{Z}_0 \\ [e^*]_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ r_1^* & wr_2^* \end{pmatrix}}_{\mathbf{M}} \cdot \begin{pmatrix} Z_{1,0} \\ Z_{2,0} \end{pmatrix} + \begin{pmatrix} 0 \\ \log_{g_1} s_b \end{pmatrix}$$

where $det(\mathbf{M}) = w(r_2^* - r_1^*) \neq 0$, provided $r_2^* \neq r_1^*$.

As soon as we fix the value of $\vec{V}$, the matrix $\mathbf{M}$ is completely fixed, but the values $Z_{1,0}$ and $Z_{2,0}$ are still uniformly and independently distributed over $\mathbb{Z}_q$. Now, fixing a value for $\mathbf{Z}_0$ also fixes a value for s_b; hence, by Lemma 96, we can conclude that the conditioned distribution of $[e^*]_3$, w.r.t. $\vec{V}$ and $\mathbf{Z}_0$, is also uniform over $\mathbb{Z}_q$. □

Lemma 99. *If $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries that A poses to the decryption algorithm, then* $\Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$.

Proof. In what follows, for $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, we will denote with $R_5^{(j)}$ the event that the j-th ciphertext $\langle i, \mathcal{B} \rangle$, submitted by $\mathcal{A}$ to the decryption oracle in game $\mathbf{G}_5$, fails to pass the test in step $D4'$, but would have passed the test in step $D4$ in game $\mathbf{G}_2$. Besides, for $1 \le j \le Q_{\mathcal{A}}(\lambda)$, we will denote with $B_5^{(j)}$ the event that the j-th ciphertext is submitted to the decryption oracle before $\mathcal{A}$ received her challenge, and with $\hat{B}_5^{(j)}$ the event that the j-th ciphertext is submitted to the decryption oracle after $\mathcal{A}$ received her challenge. If we show that, for $1 \le j \le Q_{\mathcal{A}}(\lambda)$, $\Pr[R_5^{(j)} \mid B_5^{(j)}] \le \frac{1}{q}$ and that $\Pr[R_5^{(j)} \mid \hat{B}_5^{(j)}] \le \frac{1}{q}$, then the thesis will follow.

Claim 100. $\Pr[R_5^{(j)} \mid B_5^{(j)}] \le \frac{1}{q}$.

To prove this claim, fix $1 \le j \le Q_{\mathcal{A}}(\lambda)$ and consider the quantities:

$$\vec{V} := (\mathsf{Coins}, \mathcal{H}, w, \mathbf{Z}_0, \ldots, \mathbf{Z}_v), \quad \vec{V}' := (\mathbf{X}_0, \ldots, \mathbf{X}_v, \mathbf{Y}_0, \ldots, \mathbf{Y}_v).$$

These two quantities together contain all the randomness needed to determine the behavior of $\mathcal{A}$ and of all the oracles she interacts with, up to the moment that $\mathcal{A}$ performs the encryption query. Once we fix $\vec{V}$ and $\vec{V}'$, we totally define how the adversary proceeds in her attack, before she receives her challenge back.

Moreover, fixing $\vec{V}$ and $\vec{V}'$, the event $B_5^{(j)}$ is completely defined: given $\vec{V}$ and $\vec{V}'$, we say they are *relevant*, if the event $B_5^{(j)}$ occurs. Hence, to reach the claim, it suffice to prove that the probability of event $R_5^{(j)}$, conditioned on any *relevant* values of $\vec{V}$ and $\vec{V}'$, is less then $1/q$.

Recall that the condition tested in step $D4'$ in game $\mathbf{G}_5$ is $(u_2 = u_1^w \wedge v_i = \bar{v}_i)$: since we are considering the case that the j-th query fails to pass the test in step $D4'$, but would have passed the test in step $D4$ of game $\mathbf{G}_2$, it must be the case that $t_i = \bar{t}_i$ but $u_2 \neq u_1^w$. Therefore, we only consider *relevant* values of $\vec{V}$ and $\vec{V}'$ such that $u_2 \neq u_1^w$.

Taking the logs (base g_1), the condition $u_2 \neq u_1^w$ is equivalent to $r_1 \neq r_2$ and the condition $t_i = \bar{t}_i$ is equivalent to $\beta_i = \bar{\beta}_i$, where

$$\bar{\beta}_i \doteq \log_{g_1} \bar{t}_i = r_1 X_{1,i} + w r_2 X_{2,i} + \alpha r_1 Y_{1,i} + \alpha w r_2 Y_{2,i}$$

$$\beta_i \doteq \log_{g_1} t_i = \mathsf{LI}(0, \ldots, v; \log_{g_1} t_0, \ldots, \log_{g_1} t_v)(i).$$

Notice that $\bar{\beta}_i$ can be expressed in terms of the vector

$$(X_{1,0}, X_{2,0}, \ldots, X_{1,v}, X_{2,v}, Y_{1,0}, Y_{2,0}, \ldots, Y_{1,v}, Y_{2,v})^T$$

Indeed,

$$X_{1,i} = \mathsf{LI}(0,\ldots,v; X_{1,0},\ldots,X_{1,v})(i) = \sum_{\ell=0}^{v}(X_{1,\ell}\cdot\lambda_\ell(i))$$

and similar relations hold for $X_{2,i}, Y_{1,i}$ and $Y_{2,i}$. Therefore, by means of some matrix manipulation, we can write:

$$\bar{\beta}_i = \vec{\delta}\cdot(X_{1,0}, X_{2,0},\ldots,X_{1,v},X_{2,v},Y_{1,0},Y_{2,0},\ldots,Y_{1,v},Y_{2,v})^T$$

where $\vec{\delta} \equiv (\delta_0,\delta_1,\ldots,\delta_{2v},\delta_{2v+1},\delta_{2v+2},\delta_{2v+3},\ldots,\delta_{4v+2},\delta_{4v+3})$ is defined as:

$\vec{\delta} \doteq (r_1\lambda_0(i), wr_2\lambda_0(i),\ldots,r_1\lambda_v(i),wr_2\lambda_v(i),\alpha r_1\lambda_0(i),\alpha wr_2\lambda_0(i),\ldots,\alpha r_1\lambda_v(i),\alpha wr_2\lambda_v(i))$.

In game $\mathbf{G}_5$, the random values defined above are related according to the following matrix equation:

$$\begin{pmatrix}\mathbf{X}_0\\ \vdots\\ \mathbf{X}_v\\ \mathbf{Y}_0\\ \vdots\\ \mathbf{Y}_v\\ \bar{\beta}_i\end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 & 0\\ \vdots & & & & \vdots & \vdots & & & & \vdots\\ 0 & 0 & \ldots & 1 & w & 0 & 0 & \ldots & 0 & 0\\ 0 & 0 & \ldots & 0 & 0 & 1 & w & \ldots & 0 & 0\\ \vdots & & & & \vdots & \vdots & & & & \vdots\\ 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 1 & w\\ \delta_0 & \delta_1 & \ldots & \delta_{2v} & \delta_{2v+1} & \delta_{2v+2} & \delta_{2v+3} & \ldots & \delta_{4v+2} & \delta_{4v+3}\end{pmatrix}}_{\mathbf{M}} \cdot \begin{pmatrix}X_{1,0}\\ X_{2,0}\\ \vdots\\ X_{1,v}\\ X_{2,v}\\ Y_{1,0}\\ Y_{2,0}\\ \vdots\\ Y_{1,v}\\ Y_{2,v}\end{pmatrix}$$

We want to show that the rank of the matrix $\mathbf{M}$ is $2v+3$. Clearly, the first $2v+2$ rows are linearly independent; to see why the last row (*i.e.* the vector $\vec{\delta}$) is independent from the others, notice that the only way to obtain δ_0 is by multiplying the first row by $r_1\lambda_0(i)$: doing so, the second component of δ results to be $wr_1\lambda_0(i)$. But since $\delta_1 = wr_2\lambda_0(i)$, this implies that $r_1 = r_2$, contradicting the assumption that the query fails to pass the test in step $D4'$ in game $\mathbf{G}_5$.

As soon as we fix the value of $\vec{V}$, the first $2v+2$ rows of matrix $\mathbf{M}$ are fixed, but the values $X_{1,0}, X_{2,0},\ldots,Y_{1,v},Y_{2,v}$ are still uniformly and independently distributed over $\mathbb{Z}_q$;

as for $\vec{\delta}$, its value is still undetermined, since r_1, r_2 and i are not yet fixed. Now, fixing a value for $\vec{V}'$ such that $\vec{V}$ and $\vec{V}'$ are *relevant* and that $r_1 \neq r_2$, determines the value of the j-th query (and hence the value of $\vec{\delta}$), along with the values $\mathbf{X}_0, \dots, \mathbf{X}_v, \mathbf{Y}_0, \dots, \mathbf{Y}_v$ and $\bar{\beta}_i$. Therefore, by Lemma 96, we can conclude that the distribution of $\bar{\beta}_i$, conditioned on relevant values of $\vec{V}$ and $\vec{V}'$, is uniform over $\mathbb{Z}_q$; since conditioning on any fixed, relevant value of $\vec{V}$ and $\vec{V}'$, β_i is just a single point in $\mathbb{Z}_q$, it follows that $\Pr[\beta_i = \bar{\beta}_i] = \frac{1}{q}$.

Claim 101. $\Pr[R_5^{(j)} \mid \hat{B}_5^{(j)}] \leq \frac{1}{q}$.

To prove this claim, fix $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$ and consider the quantities:

$$\vec{V} := (\mathsf{Coins}, \mathcal{H}, w, \mathbf{Z}_0, \dots, \mathbf{Z}_v, r_1^*, r_2^*, e^*), \quad \vec{V}' := (\mathbf{X}_0, \dots, \mathbf{X}_v, \mathbf{Y}_0, \dots, \mathbf{Y}_v, \beta_i^*)$$

where $\beta_i^* \doteq \log_{g_1} t_i^* = \mathsf{LI}(0, \dots, v; \log_{g_1} t_0^*, \dots, \log_{g_1} t_v^*)(i)$ and $i > v$. Notice that by the specification of the encryption oracle used in game $\mathbf{G}_5$, for $\ell = 0, \dots, v$, it holds that:

$$\log_{g_1} t_\ell^* = r_1^* X_{1,\ell} + w r_2^* X_{2,\ell} + \alpha^* r_1^* Y_{1,\ell} + \alpha^* w r_2^* Y_{2,\ell}.$$

Therefore, we can write:

$$\beta_i^* = \sum_{\ell=0}^{v} \lambda_\ell(i)(r_1^* X_{1,\ell} + w r_2^* X_{2,\ell} + \alpha^* r_1^* Y_{1,\ell} + \alpha^* w r_2^* Y_{2,\ell}).$$

Together, $\vec{V}$ and $\vec{V}'$ contain all the parameters needed to determine the behavior of $\mathcal{A}$ and of all the oracles she interacts with: once we fix $\vec{V}$ and $\vec{V}'$, we totally define how the adversary proceeds in the entire attack.

Moreover, fixing $\vec{V}$ and $\vec{V}'$, the event $\hat{B}_5^{(j)}$ is completely defined: given $\vec{V}$ and $\vec{V}'$, we say they are *relevant* if the event $\hat{B}_5^{(j)}$ occurs. Hence, to show the claim, it suffices to prove that the probability of event $R_5^{(j)}$, conditioned on any *relevant* values of $\vec{V}$ and $\vec{V}'$, is less then $1/q$.

As shown above, we can consider just relevant values of $\vec{V}$ and $\vec{V}'$ for which it holds that $u_2 \neq u_1^w$. Reasoning as in the previous case, and maintaining the notation introduced there, the random values defined above are related according to the following matrix equation:

$$\begin{pmatrix} \mathbf{X}_0 \\ \vdots \\ \mathbf{X}_v \\ \mathbf{Y}_0 \\ \vdots \\ \mathbf{Y}_v \\ \beta_i^* \\ \bar{\beta}_i \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & \dots & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & & & & \vdots & \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & w & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & w & \dots & 0 & 0 \\ \vdots & & & & \vdots & \vdots & & & & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots & 1 & w \\ \delta_0^* & \delta_1^* & \dots & \delta_{2v}^* & \delta_{2v+1}^* & \delta_{2v+2}^* & \delta_{2v+3}^* & \dots & \delta_{4v+2}^* & \delta_{4v+3}^* \\ \delta_0 & \delta_1 & \dots & \delta_{2v} & \delta_{2v+1} & \delta_{2v+2} & \delta_{2v+3} & \dots & \delta_{4v+2} & \delta_{4v+3} \end{pmatrix}}_{\mathbf{M}} \cdot \begin{pmatrix} X_{1,0} \\ X_{2,0} \\ \vdots \\ X_{1,v} \\ X_{2,v} \\ Y_{1,0} \\ Y_{2,0} \\ \vdots \\ Y_{1,v} \\ Y_{2,v} \end{pmatrix}$$

where $\vec{\delta}^* \equiv (\delta_0^*, \delta_1^*, \dots, \delta_{2v}^*, \delta_{2v+1}^*, \delta_{2v+2}^*, \delta_{2v+3}^*, \dots, \delta_{4v+2}^*, \delta_{4v+3}^*)$ is defined as $\vec{\delta}^* \doteq (r_1^*\lambda_0(i)$, $wr_2^*\lambda_0(i)$, $\dots$, $r_1^*\lambda_v(i)$, $wr_2^*\lambda_v(i)$, $\alpha^* r_1^*\lambda_0(i)$, $\alpha^* w r_2^*\lambda_0(i)$, $\dots$, $\alpha^* r_1^*\lambda_v(i)$, $\alpha^* w r_2^*\lambda_v(i))$.

We want to show that the rank of the matrix $\mathbf{M}$ is $2v+4$. Clearly, the first $2v+2$ rows of $\mathbf{M}$ are all linear independent. Moreover, as shown in the previous claim, both β_i^* and $\bar{\beta}_i$ are linearly independent from the first $2v+2$ rows of $\mathbf{M}$.

First, notice that the assumption that the j-th query $\langle i, \mathcal{B}\rangle$ is rejected in step $D4'$ of game $\mathbf{G}_5$, implies not only $\neg\Re_i(\mathcal{B}, \mathcal{B}^*)$, but also that $\mathcal{B}$ passed the *special rejection rule*; furthermore, we may assume that $\alpha \neq \alpha^*$, since otherwise the only way $\mathcal{B}$ may have passed the *special rejection rule* is that $\langle S, u_1, u_2, (j_1, H_{j_1}), \dots, (j_v, H_{j_v})\rangle = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \dots, (j_v^*, H_{j_v^*})\rangle$. But this on one hand entails $\vec{\delta}^* = \vec{\delta}$, *i.e.* $\beta_i^* = \bar{\beta}_i$, whereas on the other hand implies that $\beta_i \neq \beta_i^*$ (because otherwise $\mathcal{B}$ and $\mathcal{B}^*$ would be $\Re_i$-related). Thus, if $\alpha = \alpha^*$ then $\beta_i \neq \bar{\beta}_i$, contradicting the assumption that the j-th query $\langle i, \mathcal{B}\rangle$ would have passed the test in step $D4$ in game $\mathbf{G}_2$.

In order to show that $\vec{\delta}$ is linearly independent from the first $2v+3$ rows, observe that the only way to obtain δ_0 is by multiplying the first row by $(r_1 - r_1^*)\lambda_0(i)$ and $\vec{\delta}^*$ by 1; similarly, to obtain δ_{2v+2} as a linear combination of the other elements in its column, we need to multiply the $(v+2)$-th row by $\alpha(r_1 - r_1^*)\lambda_0(i)$ and $\vec{\delta}^*$ by $\frac{\alpha}{\alpha^*}$: since $\alpha \neq \alpha^*$,

$\frac{\alpha}{\alpha^*} \neq 1$ and so, $\vec{\delta}$ is linearly independent from all the other rows.

As soon as we fix the value of $\vec{V}$, the first $2v+2$ rows of matrix $\mathbf{M}$ are fixed, but the values $X_{1,0}, X_{2,0}, \ldots, Y_{1,v}, Y_{2,v}$ are still uniformly and independently distributed over $\mathbb{Z}_q$; as for δ^* and δ, their values are still undetermined, since r_1^*, r_2^*, r_1, r_2 and i, are not yet fixed. Now, fixing a value for $\vec{V}'$ such that $\vec{V}$ and $\vec{V}'$ are *relevant* and that $r_1 \neq r_2$, also fixes the last 2 rows of matrix $\mathbf{M}$ along with the values $\mathbf{X}_0, \ldots, \mathbf{X}_v, \mathbf{Y}_0, \ldots, \mathbf{Y}_v$ and β_i^*; hence, by Lemma 96, we can conclude that the distribution of $\bar{\beta}_i$, conditioned on relevant values of $\vec{V}$ and $\vec{V}'$, is also uniform over $\mathbb{Z}_q$; since conditioning on any fixed, relevant values of $\vec{V}$ and $\vec{V}'$, β_i is just a single point in $\mathbb{Z}_q$, it follows that $\Pr[\beta_i = \bar{\beta}_i] = \frac{1}{q}$. □

8.4.3 Proofs of Lemmas for IND-ID-CCA Security

Lemma 102. $\Pr[T_4] = \Pr[T_3]$ *and* $\Pr[R_4] = \Pr[R_3]$.

Proof. Consider the quantity $\vec{V} := (\mathsf{Coins}, \mathcal{H}, w, X_{1,0}, X_{2,0}, \ldots, X_{1,v}, X_{2,v}, Y_{1,0}, Y_{2,0}, \ldots, Y_{1,v}, Y_{2,v}, \mathbf{Z}_1, \ldots, \mathbf{Z}_v, b, r_1^*, r_2^*, k)$ and the value $\mathbf{Z}_0$. We can repeat the same considerations stated in Lemma 98: the only difference is that the quantities $(\vec{V}, \mathbf{Z}_0, [e^*]_3)$ characterizing game $\mathbf{G}_3$ are related according to the following slightly different matrix equation:

$$\begin{pmatrix} \mathbf{Z}_0 \\ [e^*]_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ r_1^* & wr_2^* \end{pmatrix}}_{\mathbf{M}} \cdot \begin{pmatrix} Z_{1,0} \\ Z_{2,0} \end{pmatrix} + \begin{pmatrix} 0 \\ \log_{g_1}(s_b \| k) \end{pmatrix}$$

For the same reasons seen in Lemma 98, as soon as we fix a value for $\vec{V}$, the matrix $\mathbf{M}$ is completely fixed, as well as the value of k, but the values $Z_{1,0}$ and $Z_{2,0}$ are still uniformly and independently distributed over $\mathbb{Z}_q$. Now, fixing a value for $\mathbf{Z}_0$ also fixes a value for s_b and hence for $\log_{g_1}(s_b \parallel k)$; thus, by Lemma 96, the conditioned distribution of $[e^*]_3$, w.r.t. $\vec{V}$ and $\mathbf{Z}_0$, is also uniform over $\mathbb{Z}_q$. □

Lemma 103. *If $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries that A poses to the decryption algorithm, then* $\Pr[R_6] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$.

Proof. In what follows, for $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, we will denote with $R_6^{(j)}$ the event that the j-th ciphertext $\langle i, \mathcal{B} \rangle$, submitted by $\mathcal{A}$ to the decryption oracle in game $\mathbf{G}_6$, fails to pass the test in step $D4'$, but would have passed both tests in step $D4$ and in step $D8$ in game $\mathbf{G}_2$. Besides, for $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, we will denote with $B_6^{(j)}$ the event that the j-th ciphertext is submitted to the decryption oracle before $\mathcal{A}$ received her challenge, and with $\hat{B}_6^{(j)}$ the event that the j-th ciphertext is submitted to the decryption oracle after $\mathcal{A}$ received her challenge. If we show that, for $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$, $\Pr[R_6^{(j)} \mid B_6^{(j)}] \leq \frac{1}{q}$ and that $\Pr[R_6^{(j)} \mid \hat{B}_6^{(j)}] \leq \frac{1}{q}$, then the thesis will follow.

Claim 104. $\Pr[R_6^{(j)} \mid B_6^{(j)}] \leq \frac{1}{q}$.

The proof of this claim closely follows the one presented in Lemma 99, so we omit the details.

Claim 105. $\Pr[R_6^{(j)} \mid \hat{B}_6^{(j)}] \leq \frac{1}{q}$.

To prove this claim we proceed like in Lemma 99, fixing $1 \leq j \leq Q_{\mathcal{A}}(\lambda)$ and considering the quantities:

$$\vec{V} := (\mathsf{Coins}, \mathcal{H}, w, \mathbf{Z}_0, \ldots, \mathbf{Z}_v, r_1^*, r_2^*, e^*), \quad \vec{V}' := (\mathbf{X}_0, \ldots, \mathbf{X}_v, \mathbf{Y}_0, \ldots, \mathbf{Y}_v, \beta_i^*, k)$$

where we are maintaining all the notations introduced above.

Again, we can repeat exactly the same construction utilized in Lemma 99: the only difference from the argument presented there is in the considerations aiming at showing that we can assume that $\alpha \neq \alpha^*$; thus, we only need to justify this assumption in the new scenario, and the claim will follow.

Under the assumptions that the j-th query $\langle i, \mathcal{B} \rangle$ is rejected in step $D4'$ of game $\mathbf{G}_6$ but would have been decrypted as valid in game $\mathbf{G}_2$, we can deduce that $\mathcal{B}$ passed both the *second special rejection rule* and the *special rejection rule*. We may also assume that $\alpha \neq \alpha^*$, since otherwise the only way that $\mathcal{B}$ may have passed the *special rejection rule* is that $\langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_v, H_{j_v}) \rangle = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_v^*, H_{j_v^*}) \rangle$; but since $\mathcal{B}$ must differ from the challenge $\mathcal{B}^*$, then it must be the case that $(t_0, \ldots, t_v) \neq (t_0^*, \ldots, t_v^*)$, and so, from the fact that $\mathcal{B}$ passed the *second special rejection rule* we get that $\tau \neq \mathsf{MAC}_{k^*}(t_0, \ldots, t_v)$, thus contradicting the assumption that the j-th query would have been decrypted in game $\mathbf{G}_2$ (since the test in step $D8$, for the validity of the tag τ, would have failed). $\square$

Bibliography

[1] S. Abdalla, M. Miner and C. Namprempre. Forward-Secure Threshold Signature Schemes. In *Topics in Cryptography—CT-RSA '01*, pages 441–456, 2001. LNCS 2020.

[2] J. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. In *Advances in Cryptology—EuroCrypt '02*, pages 83–107, Heidelberg, 2002. Springer. LNCS 2332.

[3] R. Anderson. Two Remarks on Public Key Cryptography. Invited Lecture, ACM-CCS '97. Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`, 1997.

[4] A. Barth, D. Boneh, and B. Waters. Private Encrypted Content Distribution using Private Broadcast Encryptions. In *Financial Cryptography—FC 2006*, pages ??–??, Heidelberg, 2006. Springer. LNCS ??

[5] BBCNews. Sony slated over anti-piracy CD. `http://news.bbc.co.uk/2/hi/technology/4400148.stm`.

[6] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. In *Advances in Cryptology—Crypto '99*, pages 431–448, Heidelberg, 1999. Springer. LNCS 1666.

[7] M. Bellare and B. Yee. Forward Security in Private-Key Cryptography. In *Topics in Cryptography—CT-RSA '03*, 2003. To appear. Preliminary version at `http://eprint.iacr.org/2001/035`.

[8] C. Berge. *Hypergraphs.* Elsevier, New York, 1989.

[9] O. Berkman, M. Parnas, and J. Sgall. Efficient Dynamic Traitor Tracing. In *Proceedings of the 11th Symposium on Discrete Algorithms*, pages 586–595, 2000.

[10] S. Berkovits. How to Broadcast a Secret. In *Advances in Cryptology—EuroCrypt '91*, pages 535–541, Heidelberg, 1991. Springer. LNCS 547.

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

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

[13] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal of Computing*, 13(4):850–864, 1984.

[14] C. Blundo and A. Cresti. Space Requirements for Broadcast Encryption. In *Advances in Cryptology—EuroCrypt '94*, pages 287–298, Heidelberg, 1994. Springer. LNCS 950.

[15] C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Trade-offs between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution. 387-400. In *Advances in Cryptology—Crypto '96*, pages 387–400, Heidelberg, 1996. Springer. LNCS 1109.

[16] D. Boneh, X. Boyen, and E. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Advances in Cryptology - EuroCrypt 2005*, pages 440–456, Heidelberg, 2005. Springer. LNCS 3493.

[17] D. Boneh and M. Franklin. An Efficient Public Key Traitor Tracing Scheme. In *Advances in Cryptology—Crypto '99*, pages 338–353, Heidelberg, 1999. Springer. LNCS 1666. Full version available at `crypto.stanford.edu/~dabo/pubs.html`.

[18] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology—Crypto '01*, pages 213–229, Heidelberg, 2001. Springer. LNCS 2139.

[19] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. Full version of [18].

[20] D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Advances in Cryptology - Crypto 2005*, pages 258–275, Heidelberg, 2005. Springer. LNCS 3621.

[21] D. Boneh, A. Sahai, and B. Waters. Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In *Advances in Cryptology - EuroCrypt 2006*, pages ??–??, Heidelberg, 2006. Springer. LNCS ??

[22] D. Boneh and J. Shaw. Collusion-Secure Fingerprinting for Digital Data. In *Advances in Cryptology—Crypto '95*, pages 452–465, Heidelberg, 1995. Springer. LNCS 963.

[23] D. Boneh and J. Shaw. Collusion-Secure Fingerprinting for Digital Data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998. Full version of [22].

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

[25] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-or-Nothing Transform. In *Advances in Cryptology—EuroCrypt '00*, pages 453–469, Heidelberg, 2000. Springer. LNCS 1807.

[26] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. In *Advances in Cryptology—EuroCrypt '03*, pages 255–271, Heidelberg, 2003. Springer. LNCS 2656.

[27] H. Chabanne, D. Phan, and D. Poitcheval. Public Traceability in Traitor Tracing Schemes. In *Advances in Cryptology—EuroCrypt '05*, pages 542–558, Heidelberg, 2005. Springer. LNCS 3494.

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

[29] B. Chor, A. Fiat, N. Naor, and B. Pinkas. Tracing Traitors. *IEEE Transaction on Information Theory*, 46(3):893–910, 2000.

[30] C. Cocks. An Identity-Based Encryption Scheme based on Quadratic Residuosity. In *Cryptology and Coding*, pages 360–363, Heidelberg, 2001. Springer. LNCS 2260.

[31] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology—Crypto '98*, pages 13–25, Heidelberg, 1998. Springer. LNCS 1462.

[32] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Scheme Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal of Computing*, 33(1):167–226, 2003.

[33] W. Diffie, P. van Oorschot, and W. Wiener. Authentication and Authenticated Key Exchanges. In *Designs, Codes and Cryptography*, volume 2, pages 107–125, 1992.

[34] Y. Dodis and N. Fazio. Public-Key Broadcast Encryption for Statless Receivers. In *Digital Rights Management—DRM '02*, pages 61–80, Heidelberg, 2002. Springer. LNCS 2696.

[35] Y. Dodis and N. Fazio. Public-Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. Full version of [36], available from `http://eprint.iacr.org/`, 2002.

[36] 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, Heidelberg, 2003. Springer. LNCS 2567.

[37] Y. Dodis, N. Fazio, A. Kiayias, and M. Yung. Scalable Public-Key Tracing and Revoking. In 22^{nd} *Annual Symposium on Principles of Distributed Computing—PODC '03*, pages 190–199, New York, 2003. ACM Press. Invited paper to the PODC '03 Special Issue of Journal of Distributed Computing [38].

[38] Y. Dodis, N. Fazio, A. Kiayias, and M. Yung. Scalable Public-Key Tracing and Revoking. *Journal of Distributed Computing*, 17(4):323–347, 2005.

[39] Y. Dodis and J. Katz. Chosen Ciphertext Security of Multiple Encryption. In *Theory of Cryptography—TCC '05*, pages 188–209, Heidelberg, 2005. Springer. LNCS 3378.

[40] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public-Key Cryptosystems. In *Advances in Cryptology—EuroCrypt '02*, pages 65–82, Heidelberg, 2002. Springer. LNCS 2332.

[41] N. Fazio, A. Nicolosi, and H. Phan. Traitor Tracing with Optimal Transmission Rate. In submission, 2006.

[42] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science—FOCS '87*, pages 427–437, 1987.

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

[44] A. Fiat and T. Tassa. Dynamic Traitor Tracing. *Journal of Cryptology*, 14(3):211–223, 2001.

[45] 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, Heidelberg, 1999. Springer. LNCS 1666.

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

[47] C. Günther. An Identity-Based Key Exchange Protocol. In *Advances in Cryptology—EuroCrypt '89*, pages 29–37, Heidelberg, 1989. Springer. LNCS 434.

[48] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. In *Advances in Cryptology—Asiacrypt '02*, pages 548–566, Heidelberg, 2002. Springer. LNCS 2501.

[49] M. T. Goodrech, J. Sun, and R. Tamassia. Efficient Tree-Based Revocation in Groups of Low-State Devices. In *Advances in Cryptology–Crypto '04*, pages 511–527, Heidelberg, 2004. Springer.

[50] 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.

[51] D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. In *Advances in Cryptology—Crypto '02*, pages 47–60, Heidelberg, 2002. Springer. LNCS 2442.

[52] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. In *Advances in Cryptology—EuroCrypt '02*, pages 466–481, Heidelberg, 2002. Springer. LNCS 2332.

[53] J. Hwang, D. Lee, and J. Lim. Generic Transformation for Scalable Broadcast Encryption Schemes. In *Advances in Cryptology - Crypto 2005*, pages 276–292, Heidelberg, 2005. Springer. LNCS 3621.

[54] N. Jho, J. Hwang, J. Cheon, M. Kim, D. Lee, and E. Yoo. One-Way Chain Based Broadcast Encryption Scheme. In *Advances in Cryptology—EuroCrypt '05*, pages 542–558, Heidelberg, 2005. Springer. LNCS 3494.

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

[56] A. Kiayias and M. Yung. Breaking and Repairing Asymmetric Public-Key Traitor Tracing. In *Digital Rights Management—DRM '02*, pages 32–50, Heidelberg, 2002. Springer. LNCS 2696.

[57] A. Kiayias and M. Yung. Traitor Tracing with Constant Transmission Rate. In *Advances in Cryptology—EuroCrypt '02*, pages 450–465, Heidelberg, 2002. Springer. LNCS 2332.

[58] C. Kim, Y. Hwang, and P. Lee. An Efficient Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology - Asiacrypt 2003*, pages 359–373, Heidelberg, 2003. Springer. LNCS 2894.

[59] R. Kumar, S. Rajagopalan, and A. Sahai. Coding Constructions for Blacklisting Problems without Computational Assumptions. In *Advances in Cryptology—Crypto '99*, pages 609–623, Heidelberg, 1999. Springer. LNCS 1666.

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

[61] K. Kurosawa and T. Yoshida. Linear Code Implies Public-Key Traitor Tracing. In *Public Key Cryptography—PKC '02*, pages 172–187, Heidelberg, 2002. Springer. LNCS 2274.

[62] M. Luby and J. Staddon. Combinatorial Bounds for Broadcast Encryption. In *Advances in Cryptology—EuroCrypt '98*, pages 512–526, Heidelberg, 1998. Springer. LNCS 1403.

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

[64] T. Malkin, D. Micciancio, and S. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number of Time Periods. In *Advances in Cryptology—EuroCrypt '02*, pages 400–417, Heidelberg, 2002. Springer. LNCS 2332.

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

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

[67] M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Financial Cryptography—FC 2000*, pages 1–20, Heidelberg, 2000. Springer. LNCS 1962. Full version available at `www.wisdom.weizmann.ac.il/~naor/onpub.html`.

[68] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. In L. Logrippo, editor, *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 51–60, Montréal, Québec, Canada, Aug. 1991. ACM Press.

[69] B. Pfitzmann. Trials of Traced Traitors. In *Information Hiding*, pages 49–64, Heidelberg, 1996. Springer. LNCS 1174.

[70] R. Rivest. All-or-Nothing Encryption and the Package Transform. In *Fast Softaware Encryption*, 1997.

[71] R. Safavi-Naini and Y. Wang. Sequential Traitor Tracing. In *Advances in Cryptology—Crypto 2000*, pages 316–332, Heidelberg, 2000. Springer. LNCS 1880.

[72] R. Safavi-Naini and Y. Wang. Sequential Traitor Tracing. *IEEE Transactions on Information Theory*, 49(5):1319–1326, 2003.

[73] A. Shamir. How To Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[74] A. Shamir. Identity Based Cryptosystems and Signatures Schemes. In *Advances in Cryptology—Crypto '84*, pages 47–53, Heidelberg, 1984. Springer. LNCS 196.

[75] V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption. Manuscript, 2001.

[76] G. Simmons. A Survey on Information Authentication. In *Contemporary Cryptography: The Science of Information Integrity*, pages 379–419. IEEE Press, 1992.

[77] 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.

[78] D. R. Stinson and R. Wei. Key Preassigned Traceability Schemes for Broadcast Encryption. In *Selected Areas in Cryptography*, pages 144–156, Heidelberg, 1998. Springer. LNCS 1556.

[79] G. Tardos. Optimal Probabilistic Fingerprint Codes. In *Proceedings of the 35th Symposium on Theory of Computing—STOC'03*, pages 116– 125, New York, 2003. ACM Press.

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

[81] B. Waters. Efficient Identity-Based Encryption without Random Oracles. In *Advances in Cryptology—EuroCrypt '05*, pages 114–127, Heidelberg, 2005. Springer. LNCS 3494.

[82] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In *Computer and Communications Security—CCS '04*, pages 354–363, New York, 2004. ACM Press.