MULTI-CLIENT FUNCTIONAL ENCRYPTION FOR CONTROLLED DATA SHARING

Multi-client Functional Encryption for Controlled Data Sharing

Tim Robert van de Kamp

MULTI-CLIENT FUNCTIONAL ENCRYPTION FOR CONTROLLED DATA SHARING

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. T. T. M. Palstra,
on account of the decision of the Doctorate Board,
to be publicly defended
on Friday the 21st of February 2020 at 16:45 hours

by

Tim Robert van de Kamp

born on the 23rd of October 1989
in Rotterdam, the Netherlands

This dissertation has been approved by:

Supervisor:

prof. dr. W. Jonker

Co-supervisor:

dr. A. Peter

DSI Ph.D. Thesis Series No. 19-023

ISBN: 978-90-365-4958-5

DOI: 10.3990/1.9789036549585

ISSN: 2589-7721

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Graduation Committee

Chairman/Secretary:	prof. dr. J. N. Kok
Supervisor:	prof. dr. W. Jonker
Co-supervisor:	dr. A. Peter
Committee Members:	prof. dr. F. Armknecht
	prof. dr. M. Huisman
	prof. dr. J. Müller-Quade
	prof. dr. ir. M. R. van Steen
	dr. M. H. Everts

Abstract

Multi-client functional encryption (MC-FE) is a powerful concept that makes it possible to compute on confidential data from multiple parties, while only revealing the computational outcome in the clear. This security guarantee makes MC-FE a prime candidate for controlled data sharing as long as it can reach practical efficiency for the functionalities needed in such a setting. General-purpose MC-FE schemes allow for arbitrary computations on encrypted data, but require non-standard security assumptions and are based on inefficient primitives. In contrast, special-purpose MC-FE schemes can be both efficient and proven secure under well-established assumptions. However, the few existing special-purpose schemes cover only a limited number of functionalities.

We propose special-purpose MC-FE schemes for two essential classes of data sharing functionalities: set operations and predicate testing. In the case of set operations, we construct schemes that can determine the set intersection or the cardinality thereof by employing a combination of pseudorandom functions (PRFs), hash functions, secret sharing, and elliptic curve cryptography (ECC). For the case of predicate testing, we present a compiler that turns any predicate description into an evaluation scheme operating on inputs from various parties. As a special case, we develop a construction to test the equality of vectors from multiple parties that provides a stronger security guarantee. All our predicate testing schemes are based on a combination of PRFs, hash functions, secret sharing, and bilinear maps.

We implement and evaluate several of the above schemes for sets and vectors. We see that our special-purpose schemes are at least four orders of magnitude faster than general-purpose MC-FE schemes, resulting in evaluation times of seconds on commodity hardware.

Keywords: Fine-grained data protection · Multi-client functional encryption · Non-interactive · Pairing-based cryptography

Abstract (Dutch)

Een belangwekkend begrip uit de cryptografie is *multi-client functional encryption* (MC-FE). Hiermee is het mogelijk berekeningen uit te voeren op vertrouwelijke data van meerdere partijen, terwijl enkel het resultaat van de berekening zichtbaar is. Deze beveiligingsgarantie maakt van MC-FE een geschikte kandidaat voor het gecontroleerd delen van data, zolang het maar efficiënt genoeg kan worden toegepast in de praktijk. Generieke MC-FE ontwerpen kunnen arbitraire berekeningen op versleutelde data uitvoeren, maar vereisen ongebruikelijke complexiteitsaannamen en zijn gebaseerd op inefficiënte primitieven. Specifieke MC-FE ontwerpen daarentegen, kunnen zowel efficiënt zijn alsook bewezen veilig onder gangbare complexiteitsaannamen. Echter, de specifieke MC-FE ontwerpen die bestaan, beslaan maar een klein deel van alle functionaliteiten.

Wij dragen specifieke MC-FE ontwerpen aan voor twee essentiële categorieën functies voor het delen van data: bewerkingen op verzameling en het toetsen van predicaten. In het geval van bewerkingen op verzameling maken we ontwerpen die de doorsnede of de kardinaliteit daarvan kunnen bepalen door een combinatie van blokvercijfering, hashfuncties, secret sharing en elliptische krommen toe te passen. Voor die gevallen waarbij predicaten worden getoetst, presenteren we een compiler die een omschrijving van een predicaat in een versleutelingsmethode voor dat predicaat kan omzetten. Als een bijzonder geval ontwikkelen we een constructie die betere beveiligingsgaranties kan bieden voor het testen of vectoren van meerdere partijen gelijk zijn. Al onze ontwerpen voor het toetsen van predicaten zijn gebaseerd op een combinatie van blokvercijfering, hashfuncties, secret sharing en bilineaire afbeeldingen.

We implementeren en evalueren verscheidene van de bovengenoemde ontwerpen voor verzameling en vectoren. We zien dat onze specifieke ontwerpen tenminste vier ordes van grootte sneller zijn dan generieke MC-FE ontwerpen, hetgeen blijkt uit ontcijfering in seconden op consumentenelektronica.

Acknowledgments

First and foremost, I thank the consortium partners and financiers of the CRIPTIM project. Owing to their support, I enjoyed great academic freedom and could concentrate on the cryptologic solutions that I found most interesting. I am also obliged to thank Willem Jonker and Andreas Peter for their extensive time investment and supervision during my PhD trajectory. Furthermore, it has been a pleasure to work with Maarten Everts and David Stritzl. Their contributions as co-authors are highly appreciated.

On a more personal note, I also wish to thank the many colleagues that I have met at the University. I particularly will not forget the conversations I had with Ali, Marco, and Riccardo about the ups and downs during the course of one's PhD. Also the social activities outside office hours with Bence, Chris, Herson, Roeland, Thijs, as well with Æde, Alexandr, Amina, Bertine, Claudio, Dan, Elmer, Erik, Erwin, Florian, Geert Jan, Hudi, Inés, Kemilly, Luigi, Muhammad, Philipp, Una, Valeriu, and several others were a welcome distraction. I thank you all for the pleasant time I had in the office.

Contents

1 Introduction

In our digitalized society, an ever increasing amount of data is shared by individuals, governments, and companies. Data sharing is practiced because it benefits the parties involved by providing better insights. Despite the clear advantages, most companies and organizations are also hesitant to share their data. They realize that the moment they share their data, they lose their ability to control how their data is being used. Once the data is shared, it could be used by others (with potentially less benign intentions) to infer more information than originally intended by the data sharer. Events from the past have shown that the additional information that can be inferred from shared data can be quite surprising [BZ06; GP09; G0013; Tr014], indicating that it is hard to predict all potential negative consequences of data sharing.

1.1 Controlled Data Sharing

In an effort to mitigate the risks of losing control over shared data, several approaches exists. Since data is like any other asset a company might possess, the decision on whether a specific data protection technique is appropriate depends on the sensitivity of the data and the impact of losing control over the data. Hence, governments and companies commonly take a combination of three approaches to minimize the risk arising from data sharing: limit the sharing to trusted partners, share within a legal framework, and minimize the data that needs to be shared.

Trust-based approaches try to foster the establishment of trust among parties. A strong governmental push [PCC97; IRTPA; EULEX] for information sharing has led to several trust-based sharing initiatives. For example, in Europe, representatives of several critical infrastructure (CI) companies exchange sensitive information during physical meetings organized by information sharing and analysis centers (ISACS). Data is only shared with another party if that party is trusted to act in accordance with the rules on how the data may be processed. Typically, these rules include restrictions on how the information may be disseminated, *e.g.*, using the traffic light protocol [LK15a]. A major drawback of trust-based solutions is that trust is slow to establish, not transitive, and often personal, *i.e.*, not the company is trusted, but a person working at that company. For these reasons, trust-based solutions do

not scale well.

Using a *legal framework*, *e.g.*, by using contracts and legislation, parties can formalize their agreement on how shared data should be processed. Additionally, parties can decide to set penalties on the improper processing of data or the misuse of the shared information. While parties may hope to discourage data misuse using the repressive effect from imposing big fines on the misuse of data, legal enforcement cannot *prevent* data misuse: Legal enforcement cannot control how shared data is used by other parties, it can only enforce rules whenever a party already committed a breach of contract.

To prevent data misuse, a simple solution is to not share the data at all. Of course, this is not an option if parties want to seize the benefits of information sharing. Instead, parties can apply *data minimization* to share only the data that is needed to achieve a predefined goal. For example, if it is only needed to get an impression of a data set, parties can share statistics that summarize the data set, instead of revealing the individual data items. However, it is not always possible to define the precise goals of the data sharing upfront. Even worse, it is not always so simple to solely reveal the needed information without indirectly revealing more than intended. For example, consider an auction where the seller needs to know who won the bidding, but the bidders do not want to reveal their bid if they have not won.

A combination of the approaches described above can help to achieve more controlled data sharing through the use of a trusted third party (TTP). In such a case, several parties only reveal their data to a common TTP, each participant entrusting the third party to carry out a computation on their data and to only reveal the outcome of that computation. This way, the amount of data that is revealed to parties other than the TTP can be minimized, as they only need to learn the output of the computation, not the inputs to the computation.

1.2 Controlled Data Sharing By Fine-Grained Data Protection

Using data protection techniques, we can prevent unauthorized parties from gaining access to data and thus limit the possibilities of data misuse. We can therefore regard data protection as a method to implement data minimization in the context of data sharing. However, the use of traditional data protection techniques, such as symmetric and public key encryption, do not allow for *fine-grained* data protection nor data sharing. For example, consider a scenario where we only want to share information about data items that we have in common with another party. Using traditional encryption we cannot solve this problem: We either need to reveal all our data items to the other party,

the other party needs to reveal all their data items to us, or we need to rely on a trusted third party to tell us which data items we have in common.

The limitation of traditional encryption comes from the fact that we are required to remove the encryption (and thereby also the protection mechanism) before we can meaningfully share the data with another party. In contrast, modern cryptographic techniques enable parties to share their data cryptographically protected, while still allowing other parties to infer information from the data via restricted computations on the encrypted data. With the same techniques we can choose to disclose only part of our data or only disclose information if certain conditions have been met.

While we could already share a minimized form of data with other parties using the help of a trusted third party, a cryptographic approach does not suffer from having to trust another party. Using cryptography, we can prove to preserve the confidentiality of the data that parties do not want to share with anyone under a well-specified attacker model and complexity assumptions.

1.3 Research Objective

Considering that fine-grained data protection can help to better control data sharing, we set ourselves the objective to develop effective data protection techniques for fine-grained data sharing. Our research objective can therefore be summarized as follows.

Research Objective

> "Develop effective fine-grained data protection techniques that enable parties to only reveal the information that they want to share."

As explained in the section above, computations on encrypted data enable us to selectively reveal information. Several cryptographic techniques are capable of computing on encrypted data. However, we set our objective to develop an *effective* technique. We therefore restrict ourselves to practical techniques that minimize our assumptions about the settings in which data is shared.

To minimize the number of assumptions, we avoid constraining our solutions to situations where all parties need to be on-line during data sharing. We do this because solutions that use interactive protocols are unsuitable in situations where communication is costly (*e.g.*, when using resource constrained devices) or where communication is impossible (*e.g.*, if not all participants are aware of each other's participation). This means that we exclude cryptographic techniques such as secure multi-party computation, since they

require multiple rounds of communication between the participants. Similarly, we also have to exclude homomorphic encryption (HE) techniques: While HE allows for computations on encrypted data, the output of the computation is still encrypted. Therefore, for the shared data to be useful, we require at least one round of communication to learn the decrypted output of the computation. Instead, using functional encryption (FE) we can avoid interaction while still allowing for computations on encrypted data. In an FE scheme, decryption keys are associated with a functionality f and the decryption of an encrypted message m returns the function applied to the message, $f(m)$. As a result, we primarily consider FE schemes.

To develop practical techniques, we construct and evaluate FE schemes for functionalities useful for generic data sharing. That is, we do not construct new encryption schemes aimed at solving a specific problem, instead, we construct schemes that can be used as a building block in many data sharing settings. While FE schemes for generic functionalities exists [GGH+13a; GGH+13b; GKP+13], they rely on heavy cryptographic machinery such as indistinguishability obfuscation ($i\mathcal{O}$) using multilinear maps. Because these constructions are slow [AHK+14; BOK+15; HHS+17] and based on primitives which have already been broken several times [CHL+15; CLL+16; CGH17; BHJ+19], we consider FE schemes for smaller function classes. By limiting FE to a small class of functions or even to a single functionality, we aim to develop practically efficient techniques for data sharing.

To analyze whether an FE type is suitable for effective data sharing, we first need to say something about the type of data sharing. We can classify data sharing by looking at the number of senders and receivers. For example, a face-to-face meeting is a form of *one-to-one* data sharing, while posting an article on a website is of the form *one-to-many*. There can also be multiple data senders. For example, in *many-to-one* data sharing, several parties share their data with a single party.

We find that non-interactive techniques for the one-to-x type of data sharing can be very efficient [KPE+16], but also that such techniques can never be secure when part of the data that needs to remain secret is predictable (*i.e.*, having a low *min-entropy*). The reason for this limitation of non-interactive one-to-x data sharing is simple to explain through an example. Suppose Alice wants to share a message m with Bob, if and only if predicate P evaluates to TRUE based on Bob's data items. However, even if Alice encrypts both her message and predicate, Bob is able to learn the message m if he can predict which data items make the predicate evaluate to TRUE. To do so, Bob can just evaluate the encrypted predicate for many possible inputs (even for inputs other than his own data items) until he finds the input that makes

the predicate evaluate to TRUE. Since no interaction is required, Bob can evaluate the predicate many times—even without Alice noticing—and, by the functionality that the construction provides, Bob is able to learn the value m if he knows an input that evaluates the predicate to TRUE.

Generalizing the example, we see that if Alice wants to disclose the value $f(m_A, m_B)$ based on Bob's input m_B in a non-interactive way, Alice has to share the function $f_{m_A}(\cdot) = f(m_A, \cdot)$. However, by this provided functionality, Bob can determine the values $f_{m_B}(m_A) = f(m_A, m_B)$ for many different inputs m_B and learn as much as possible about Alice's input m_A.

To avoid the inherent negative security result from one-to-x type of data sharing, we look at the more interesting case of many-to-x data sharing. We develop constructions for the type of data sharing where a function f is evaluated on multiple inputs from different parties. This concept corresponds to multi-client functional encryption (MC-FE) where the decryption algorithm requires a decryption key, associated with an n-ary function f, and n encrypted values $x_1, \ldots, x_n$ to output $f(x_1, \ldots, x_n)$. Using MC-FE it is possible to compute a function on the data items of all sharing participants, while also guaranteeing that the parties' individual data items remain confidential. An example MC-FE used in the context of fine-grained information sharing can be found in sharing the sum of several participants inputs, while keeping each participant input confidential [KPE+16].

Based on the discussion above, we formulate our first research question for constructing MC-FE schemes for basic data sharing functionalities.

Research Question 1

> "How to construct fine-grained multi-client functional encryption schemes for basic functionalities, such as set intersections and predicate evaluations?"

Besides constructing MC-FE for new functionalities, we also want to assess the practically of the constructions. Whether a solution is practical, depends on the application, effectiveness, the functionality, and the efficiency of the implementation. Since we are concerned with functionalities for generic use cases, we can simplify the practicability assessment to the following question:

Research Question 2

> "How efficient can the use of multi-client functional encryption in the context of data sharing be?"

We assess the efficiency of the schemes by creating proof-of-concept implementations and evaluation our implementations on commodity hardware.

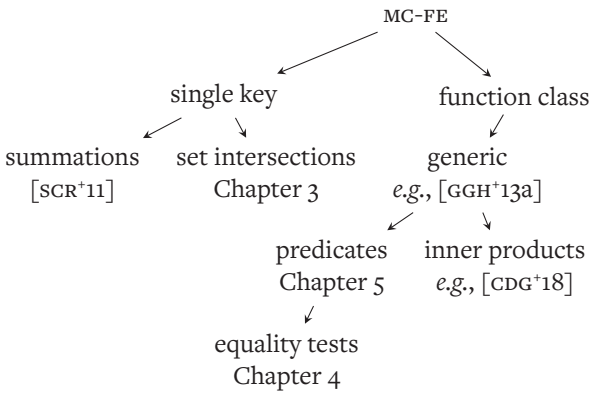


Figure 1.1. Hierarchy of MC-FE schemes for various functionalities.

However, the decision whether a scheme is efficient enough in practice, heavily depends on the use case. For example, an encryption scheme that takes one second to decrypt a message is not suitable for a system that monitors live network traffic, while the same scheme might be perfectly suitable in a case where data is shared on an hourly basis. The evaluations we provide in this dissertation should merely give an impression on whether the use of a certain technique is viable for a specific setting.

1.4 Dissertation Outline

After discussing the preliminaries on MC-FE in Chapter 2, containing the definitions of the attacker models and the often used complexity assumptions, we look at several MC-FE constructions. In Chapters 3 to 5, we discuss several MC-FE schemes grouped by the provided function classes. Chapter 6 contains several directions for extending the work before we conclude in Chapter 7.

Figure 1.1 gives a schematic overview of how our MC-FE constructions relate to MC-FE constructions that have been proposed before. We distinguish two types of MC-FE: schemes that only support one functionality—also termed *single key* [KLM+18], since there is only one decryption key for a single functionality—and schemes that support a class of functions.

1.5 Contribution

To answer both research questions, we construct and evaluate several MC-FE schemes for various functionalities.

Set Intersections For situations where parties only want to selectively reveal their data, we construct MC-FE schemes for set intersections in Chapter 3.

Using such a scheme parties can decide to only share how many data items they have in common with other parties or only share information about those data items they have in common with other parties. This functionality allows for privacy-preserving profiling where, for example, law enforcement only learns the identity of individuals who match all search criteria.

We show that it is possible to construct very efficient two-client FE schemes using already widely used cryptographic primitives. While the multi-client counterparts for many-to-one data sharing require more advanced cryptographic constructions, we are the first to show the existence of such a scheme. Additionally, in Chapter 6, we detail an idea for new constructions both improving the evaluations complexity and satisfying more stringent security guarantees.

Equality Tests Instead of constructing MC-FE for one specific functionality, we construct an MC-FE scheme for a class of equality testing functionalities in Chapter 4. By choosing to construct MC-FE for a class of functionalities, we do not require parties to know precisely what information they have to reveal: By changing the decryption key, we can change the specific functionality that is evaluated.

We motivate our construction by applying it as a monitoring system in the context of critical infrastructure protection. A remarkable feature of the proposed construction is that neither the functionality, nor the participants' data have to be revealed in order to monitor the participants. Instead, the only information that is disclosed to the monitor is whether it should raise an alarm or not. By implementing our scheme, we see that in a realistically sized monitoring system, evaluations of the functionality can be done in well under a second.

General Predicates In an effort to allow for an even larger functionality class that is still relatively efficient to evaluate, we consider the class of general predicates. In Chapter 5, we extend the concept of a pair encoding scheme to construct a variant for multiple parties, a multi-authority admissible pair encoding scheme (MA-PES). Using such an MA-PES for a specific predicate, we obtain a multi-authority predicate encryption (MA-PE) scheme by applying a conversion algorithm. We propose MA-PESS for identity-based encryption, attribute-based encryption, and inner-product predicate encryption (IPPE), achieving multi-authority versions of these predicate encryption types. Our generic approach allows us, for the first time, to construct multi-authority schemes that can consist of different predicate types. Moreover, we show how to construct the first multi-authority IPPE scheme.

A big practical advantage of our MA-PE construction is that the resulting schemes avoid key escrow by a single authority and do not require authorities to be aware of each other's existence. Concretely, this allows a content provider to broadcast encrypted content while ensuring that only its costumers with a matching profile (*e.g.*, "student" or "18+") can decrypt the content. Such a profile can combine various attributes from distinct authorities, *e.g.*, a "student" attribute issued by a university and an "18+" attribute issued by a local government.

Additionally, in Chapter 6, we detail an idea for basing our construction on prime-order bilinear groups, leading to a faster scheme for the same security level.

2 Preliminaries

In this chapter, we give an overview of the notations, used primitives, complexity assumptions, and other definitions. Some of the later chapters also contain a preliminaries section, there we only cover the preliminaries needed for that specific chapter. Depending on the background of the reader, this chapter may be skipped and only used as a reference once something is unclear in one of the following chapters.

Throughout this dissertation, we use $x \xleftarrow{R} \mathcal{S}$ to denote that x is chosen uniformly at random from the finite set $\mathcal{S}$. If an element $x \in \mathcal{S}$ is a uniformly random element from the finite set $\mathcal{S}$, we write $x \in_R \mathcal{S}$. The ordered set of number $\{1, \dots, n\}$ is denoted by $[n]$, while we denote the ordered set $\{0, \dots, n\}$ by $[n]^+$. We denote the ith component of a vector $\mathbf{v}$ as v_i. For a set of indices $\mathcal{I}$, we write $\mathbf{v}_\mathcal{I}$ for the subvector of $\mathbf{v}$. Instead of consistently using the vector notation, we use the set notation when this is clearer or more convenient. For example, use a derived notation for n-ary functions to denote that some of its (not necessarily consecutive) inputs are hardwired in the function as $f(\{x_i\}_{i \in \mathcal{I} \subset [n]}, \cdot)$. We work with notation for vectors of group elements, *e.g.*, the column vector $(g^{v_1}, \dots, g^{v_n})^\mathsf{T}$ is written as $g^\mathbf{v}$. For matrices we use upper case variables such as M. In the descriptions of the security games, we use the standard notation $\mathcal{A}^{\mathcal{O}(\cdot)}$ for an adversary $\mathcal{A}$ having oracle access to a subroutine $\mathcal{O}$ without knowing the specification of this subroutine other than its domain and codomain. We denote computational indistinguishability using the binary relation $\approx_c$.

2.1 Common Primitives

We use several symmetric-key primitives for encryption.

We (informally) say that a function F_k is a pseudorandom function (PRF) if the output of F_k is indistinguishable from the output of a function R chosen uniformly at random from the set of all possible functions from $\mathcal{M}$ to $\mathcal{C}$.

Definition 1 (Pseudorandom Function)**.** A family of functions with domain $\mathcal{M}$ and codomain $\mathcal{C}$ indexed by a key $k \in \mathcal{K}$, *i.e.*, $\{F_k\}_{k \in \mathcal{K}}$ with $F_k \colon \mathcal{M} \to$

C, is pseudorandom, if for any probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$, its advantage in distinguishing F_k from a uniformly at random sampled function R,

$$\left|\Pr_k\left[\mathcal{A}^{F_k(\cdot)}=1\right]-\Pr_{R\colon \mathcal{M}\to\mathcal{C}}\left[\mathcal{A}^{R(\cdot)}=1\right]\right|,$$

is negligible.

Similarly, a function P_k is (informally) termed a pseudorandom permutation (PRP) if the output of P_k is indistinguishable from the output of a permutation chosen uniformly at random from the set of all possible permutations over $\mathcal{M}$. The advanced encryption standard (AES) is a well-known example of a PRP.

Definition 2 (Pseudorandom Permutation)**.** A family of permutations over the domain $\mathcal{M}$ indexed by a key $k \in \mathcal{K}$, *i.e.*, $\{P_k\}_{k\in\mathcal{K}}$ with $P_k\colon \mathcal{M}\to\mathcal{M}$, is pseudorandom, if for any p.p.t. adversary $\mathcal{A}$, its advantage in distinguishing P_k from a uniformly at random sampled permutation π,

$$\left|\Pr_k\left[\mathcal{A}^{P_k(\cdot),P_k^{-1}(\cdot)}=1\right]-\Pr_{\pi\colon \mathcal{M}\to\mathcal{M}}\left[\mathcal{A}^{\pi(\cdot),\pi^{-1}(\cdot)}=1\right]\right|,$$

is negligible.

Another primitive we use is secret sharing.

Definition 3 (Shamir's Secret Sharing Scheme [Sha79])**.** The t-OUT-OF-n Shamir's secret sharing scheme uses a t-degree polynomial f over a finite field $\mathbb{F}_p$. To share a secret s, set $f(0)=s$ and pick all other coefficients of the polynomial uniformly at random in $\mathbb{F}_p$. Shares can be created by picking a tuple $(i, f(i))$ for distinct values i. To recover the secret from a set containing at least t distinct shares $\{\,(i,f(i)) : i\in\mathcal{S}, |\mathcal{S}|\geq t\,\}$, Lagrange interpolation is used, $f(0)=\sum_{i\in\mathcal{S}} f(i)\cdot\Delta_{\mathcal{S},i}$, where $\Delta_{\mathcal{S},i}=\prod_{j\in\mathcal{S},j\neq i}(j\cdot(j-i)^{-1})$.

In our functional encryption (FE) schemes we make extensive use of non-degenerate *bilinear maps*, also termed *pairings*.

Definition 4 (Pairing)**.** Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic multiplicative groups of finite order N. The map $e\colon \mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$ is a *pairing* if the following two conditions hold:

- The map is bilinear; for all generators $g_1\in\mathbb{G}_1$, $g_2\in\mathbb{G}_2$, and $a,b\in\mathbb{Z}_N$, we have that $e(g_1^a,g_2^b)=e(g_1,g_2)^{ab}$.

- The map is non-degenerate; there exists generators g_1 and g_2 such that the order of the element $e(g_1, g_2) \in \mathbb{G}_T$ equals N, the order of group $\mathbb{G}_T$.

Following the classification of Galbraith, Paterson, and Smart [GPS08], we distinguish three types of pairings:

Type 1 a *symmetric* pairing with $\mathbb{G}_1 = \mathbb{G}_2$;

Type 2 an *asymmetric* pairing with $\mathbb{G}_1 \neq \mathbb{G}_2$ and an efficiently computable homomorphism $\phi\colon \mathbb{G}_2 \to \mathbb{G}_1$;

Type 3 an *asymmetric* pairing with $\mathbb{G}_1 \neq \mathbb{G}_2$, but without a known efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

We use the function $\mathcal{G}_i(1^\lambda)$ to generate the parameters for a Type i pairing for security parameter λ.

Type 1 pairings—constructed using supersingular elliptic curves—can have a composite order N [BGN05], while only pairings of prime-order groups are known for Type 2 and Type 3 pairings. In these composite-order pairings with $N = p_1 p_2 \cdots p_m$, we refer to the subgroups of $\mathbb{G}$ of prime order p_1, p_2, …, p_m, as $\mathbb{G}_1$ till $\mathbb{G}_m$, respectively. Similarly, we write g_1, …, g_m for the generators of the respective subgroups.

These composite-order pairings possess the useful *orthogonality* property [Lew12] that fulfills a crucial role in the security proofs of schemes that use composite-order pairings. Let g denote a generator of the group of order $N = p_1 p_2 \cdots p_m$. For two generators $g_i = g^{N/p_i}$ and $g_j = g^{N/p_j}$ of distinct subgroups of order p_i and p_j (*i.e.*, $p_i \neq p_j$), we have the following property

$$e(g_i, g_j) = e\Big(g^{N/p_i}, g^{N/p_j}\Big) = e(g, g)^{N/p_i \cdot N/p_j} = \mathbb{1}.$$

Recent advancements in the field of Index Calculus have important consequences for the security of pairing-based cryptography. For example, recent results [BD18] show that a 254 bit BN curve (Type 3) with an embedding degree of 12 [BNO6], has a bit security level of about 100 bits. For lower embedding degrees the attacks are even more devastating: both pairings using a 512 bit supersingular curve $y^2 = x^3 + x$ (Type 1) with embedding degree of 2 and a 159 bit MNT curve (Type 3) with embedding degree of 6 [MNT01] only have a bit security level of about 60 bits. However, new pairings are proposed [FM19] that are secure against these attacks.

2.2 Complexity Assumptions

We use a variety of complexity assumptions in our constructions. Assumptions specific to a construction, we cover in the preliminaries of the chapter specific to the construction.

Assumption 1. The decisional Diffie–Hellman (DDH) assumption states that, given $(\mathbb{G}, g \in \mathbb{G}, g^a, g^b, T)$ for uniformly at random chosen a and b, it is hard to distinguish $T = g^{ab}$ from $T \xleftarrow{R} \mathbb{G}$.

The DDH assumption is easily broken if a pairing on $\mathbb{G}$ is available. Therefore, when using pairing-based cryptography, we often need other assumptions as well.

The d-Linear assumption is a generalization of the decision linear assumption [BBS04] and is also defined in pairing groups [BW06; Sha07]. Note that the 1-Linear assumption is the DDH assumption (which does not hold in pairing groups) and that the 2-Linear assumption is the decision linear assumption in pairing groups.

Assumption 2 (d-Linear [BBS04; BW06; Sha07])**.** Let $\mathsf{gp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ be generated by $\mathcal{G}_3(1^\lambda)$. We define $\mathbf{a} \xleftarrow{R} (\mathbb{Z}_p^*)^d, a_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$ and $\mathbf{s} \xleftarrow{R} (\mathbb{Z}_p)^d, s_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$. Given $(\mathsf{gp}, g_1^{\mathbf{a}}, g_1^{a_{d+1}}, g_2^{\mathbf{a}}, g_1^{\mathbf{a} \circ \mathbf{s}}, T)$, where $\circ$ denotes the pointwise product, it is hard to distinguish if $T = g_1^{a_{d+1} \sum_{i=1}^{d} s_i}$ or $T = g_1^{a_{d+1} \sum_{i=1}^{d} s_i + s_{d+1}}$.

Another complexity assumption that typically occurs in pairing-based cryptography is the symmetric external Diffie–Hellman (SXDH) assumption.

Assumption 3 (Symmetric External Diffie–Hellman)**.** Given the bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the symmetric external Diffie–Hellman (SXDH) assumption states that the DDH problem in both group $\mathbb{G}_1$ and group $\mathbb{G}_2$ is hard.

If the DDH problem is only required to hold in one of the groups $\mathbb{G}_1$ or $\mathbb{G}_2$, we refer to this assumption as the external decisional Diffie–Hellman (XDDH) assumption.

2.3 Definitions of Functional Encryption Schemes

In an FE scheme [BSW11; ONe10], decryption keys are associated with a functionality f and the decryption of an encrypted message m returns the function applied to the message, $f(m)$, instead of the original message m. This concept can be extended to functions with more than one input, resulting in a

multi-input functional encryption (MI-FE) scheme [GGG+14]. Correspondingly, the decryption algorithm of an MI-FE scheme requires a decryption key, associated with an n-ary function f, and n encrypted values $x_1, \dots, x_n$ to output $f(x_1, \dots, x_n)$.

A strict subset of these MI-FE schemes are termed multi-*client* functional encryption (MC-FE) schemes [GGG+14]. In such an MC-FE scheme, the inputs for the n-ary function f are given by n distinct parties, termed *clients*. Each client encrypts their input using their own encryption key, usk, and a *time-step* or *session identifier*, ID. This identifier is used to determine which ciphertexts from the various clients belong together. To evaluate a function f using the corresponding decryption key, all inputted ciphertexts need to be associated with the same identifier or otherwise decryption will fail.

Definition 5 (Multi-client Functional Encryption)**.** An MC-FE scheme for a functions class $\mathcal{F}$ of polynomial time functions, consists of the following four polynomial time algorithms.

Setup$(1^\lambda, n) \to (\mathsf{pp}, \mathsf{msk}, \mathsf{usk}_1, \dots, \mathsf{usk}_n)$**.** On input of the security parameter λ and the number of clients n, the algorithm outputs the public parameters pp, a master secret key msk, and the clients' secret keys usk_i for each client $1 \leq i \leq n$.

The public parameters pp are implicitly used by the other algorithms.

KeyGen$(\mathsf{msk}, f) \to \mathsf{esk}_f$**.** The key generation algorithm is used to create an evaluation key for the function f. To create such an evaluation key esk_f, the algorithm takes the master key msk and a description of a function $f \in \mathcal{F}$ as input.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, x_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** For a client i to encrypt a value x_i for identifier ID, the client uses its secret key usk_i and outputs the ciphertext $\mathsf{ct}_{\mathsf{ID},i}$.

Eval$(\mathsf{esk}_f, \mathsf{ct}_{\mathsf{ID},1}, \dots, \mathsf{ct}_{\mathsf{ID},n}) \to f(x_1, \dots, x_n)$**.** An evaluator having the evaluation key esk_f and a ciphertext for identifier ID from every client, outputs the function evaluation $f(x_1, \dots, x_n)$.

Note that we informally defined correctness in the Eval algorithm of Definition 5. Similarly, in the following chapters, we show correctness of a scheme inside the definition of the Eval (or Decrypt) algorithm if the proof is relatively simple.

2.4 Security Definitions

In this dissertation we use indistinguishability-based security games. Many, slightly different, security definitions exists for MC-FE. Typically, we at least require that the encrypted clients' values $\mathsf{ct}_{\mathsf{ID},i}$ do not reveal information about

the plaintext values x_i (sometimes referred to as *plaintext-privacy* [SSW09]). We might also require that an evaluation key esk_f does not reveal which specific functionality f from the class of functions $\mathcal{F}$ it is associated with (this notion is termed *predicate-privacy* [SSW09]).

Besides the question of what needs to be hidden from the adversary, different security games exists for under which conditions security is proven to hold. For example, we often require that even if a subset of the clients $\bar{I} \subset [n]$ is corrupt, *i.e.*, share their usk with the adversary, the ciphertexts from the uncorrupted clients $I = [n] \setminus \bar{I}$ are still indistinguishable from random values. Also, as a more technical example, we can require the adversary in a security game to commit to (part of) its challenge options before receiving the scheme's public parameters. Such type of security game is termed *selective*. A selective game can be turned into an adaptive security game using complexity leveraging, *i.e.*, by letting the adversary guess the challenge options, at the costs of an exponential security loss [BB04a, Theorem 7.1].

One of the strongest security games for MC-FE are termed *full security under static corruptions*, also referred to as "adaptive IND-security" in the symmetric-key setting [GKL+13, Definition 2.6]. This game satisfies plaintext-privacy if the corrupted clients are announced by the adversary before seeing the public parameters.

Definition 6 (Adaptive IND-security of MC-FE)**.** An MC-FE scheme is *fully secure under static corruptions* if any p.p.t. adversary $\mathcal{A}$ has at most a negligible advantage in winning the following game.

Corruptions The adversary sends a set of uncorrupted and corrupted clients to the challenger, I and $\bar{I}$, respectively.

Setup The challenger $\mathcal{B}$ picks a bit $b \xleftarrow{R} \{0,1\}$, and sends the public parameters pp along with the user keys of the corrupted clients $\{\mathsf{usk}_i\}_{i \in \bar{I}}$ to the adversary $\mathcal{A}$.

Query 1 The adversary may query the challenger for the encryption of values x_i for uncorrupted clients $i \in I$ associated with an ID that has not been used before. For each uncorrupted client $i \in I$, the challenger returns the encrypted value $\mathsf{ct}_{\mathsf{ID},i} \leftarrow \mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{ID}, x_i)$.

Additionally, the adversary may query for evaluation keys by submitting a function $f \in \mathcal{F}$ to the challenger. The challenger responds using $\mathsf{esk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$.

Challenge The adversary sends two equally sized values $x^*_{i,0}$ and $x^*_{i,1}$ for every uncorrupted client $i \in I$ together with an ID^* that has not been used before. The challenger checks if the challenge is allowed by checking if $f(\{x^*_{i,0}\}_{i \in I}, \cdot) = f(\{x^*_{i,1}\}_{i \in I}, \cdot)$ for all queried f. If this is not the

case the challenger aborts the game. Otherwise, it returns the ciphertexts $\mathsf{ct}^*_{\mathsf{ID}^*,i} \leftarrow \mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{ID}^*, x^*_{i,b})$ for every uncorrupted client $i \in I$.

Query 2 Identical to Query 1, with the additional restriction that new key queries must not violate the constraint described in Challenge.

Guess The adversary $\mathcal{A}$ outputs its guess b' for the challenger's bit b. We say that $\mathcal{A}$ wins the game if $b' = b$.

3 Set Intersections

Two-Client and Multi-client Constructions

As explained in the introduction, we construct various multi-client functional encryption schemes for different functionalities. In this chapter we construct multiple encryption schemes for set intersection and variants on two or more sets, thereby addressing part of Research Question 1. We are the first to consider several set operations in a non-interactive setting such as multi-client functional encryption. In our schemes, a party may learn a set operation (*e.g.*, set intersection) from the sets of two or multiple clients, without having to learn the plaintext set of each individual client. For the case of two clients, we construct efficient schemes for determining the set intersection and the cardinality of the intersection. To evaluate the cardinality of the intersection, no overhead is incurred compared with operating on plaintext data. We also present other functionalities with a scheme for set intersection with data transfer and a threshold scheme that only discloses the intersection if both clients have at least t elements in common. Finally, we consider set intersection and set intersection cardinality schemes for the case of three or more clients from a theoretical perspective. Answering Research Question 2 using our proof-of-concept implementations, we show that the two-client constructions are very efficient—running in less than milliseconds—and scale linearly in the set sizes. The multi-client constructions are more involved and can be evaluated in the order of seconds.

This chapter is based on the work "Two-Client and Multi-client Functional Encryption for Set Intersection" [KSJ+19], presented at ACISP.

3.1 Introduction

In this chapter, we explore the set intersection functionality and several variants. Inspired by the popularity of private set intersection (PSI) pro-

functionality	two-client	multi-client
set intersection	§ 3.6.2	§ 3.7.3
set intersection cardinality	§ 3.6.1	§§ 3.7.1, 3.7.2
set intersection with data transfer	§ 3.6.3	open problem (see Chapter 6)
threshold set intersection	§ 3.6.4	open problem

Table 3.1. Overview of the presented MC-FE schemes for set operations.

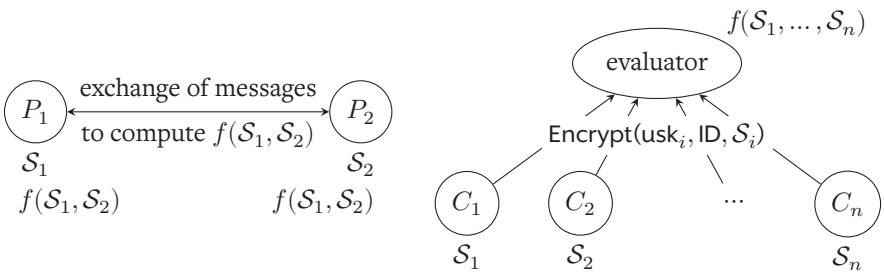
tocols [PSZ14], we define a scheme for determining the set intersection of two clients' sets in a *non-interactive* manner. Additionally, we propose several other non-interactive variants of interactive PSI protocols that were previously proposed in literature. We construct a two-client functional encryption (2C-FE) scheme for determining the cardinality of the intersection (*i.e.*, $|\mathcal{S}_a \cap \mathcal{S}_b|$, where $\mathcal{S}_\gamma$ is the set belonging to client γ), similar to PSI cardinality [KS05]. We also consider a non-interactive 2C-FE version of the less common PSI with data transfer [DT10; JL10], where the common set elements are shared with associated data (*i.e.*, $\{ (x_j, \varphi_a(x_j), \varphi_b(x_j)) \mid x_j \in \mathcal{S}_a \cap \mathcal{S}_b \}$, where $\varphi_\gamma(x_j)$ is the data associated with x_j by client γ). Finally, we construct a threshold scheme where the set intersection is only revealed if two clients have at least t set elements in common.

Following our 2C-FE schemes, we also explore the much harder multi-client case where we propose multi-client functional encryption (MC-FE) schemes for determining the (cardinality of the) set intersection of more than two sets. While 2C-FE schemes could also be used to determine the intersection of multiple sets, doing so would leak information about the intersection of each pair of sets. To prevent this undesirable leakage and achieve secure MC-FE for set intersection, we require more involved constructions.

An overview of constructions for MC-FE for set intersection presented in this work is given in Table 3.1.

Although the functionalities for our MC-FE schemes are inspired by various PSI protocols, the usage scenario differs in a crucial way: We apply our MC-FE schemes in a scenario where a third party, termed the *evaluator*, learns the function outcome. In Section 3.5.1 we explain why non-interactive 2C-FE cannot be secure if one of the clients also serves as the evaluator. We highlight the difference between PSI and our MC-FE for set intersection in Figure 3.1.

Using the functionalities provided by our constructions, it is possible to achieve privacy-preserving profiling. For example, consider a case where the police is looking for suspects which were both present at a concert and recently received a large sum of money on their bank account. Using a 2C-FE scheme for determining the set intersection, the police will only learn about



(a) A typical scenario of PSI. Both parties learn the output of the function evaluation, but not each other's inputs.

(b) Our scenario of MC-FE for set intersection. The evaluator learns the function evaluation and nothing else about the clients' inputs.

Figure 3.1. Fundamental difference between a private set intersection (PSI) protocol and our multi-client functional encryption (MC-FE) schemes for set intersection.

the suspects matching the two profiles, while learning nothing about the other visitors of the concert or other people that received an unusual amount of money. Another use case is privacy-preserving data mining, such as the computation of various set similarity scores. For example, by determining the cardinality of a set intersection we can compute the Jaccard index (*i.e.*, $|\mathcal{S}_1 \cap \mathcal{S}_2| / |\mathcal{S}_1 \cup \mathcal{S}_2| = |\mathcal{S}_1 \cap \mathcal{S}_2| / (|\mathcal{S}_1| + |\mathcal{S}_2| - |\mathcal{S}_1 \cap \mathcal{S}_2|)$), without requiring the evaluator to learn the clients' sets themselves. Finally, for the use case of privately sharing context of an indicator of compromise [CFS+17], our construction for set intersection with data transfer or projection can be used.

To asses the practicability of our constructions, we implemented several of our proposed schemes. Our 2C-FE constructions are quite efficient: Determining the cardinality of the set intersection of two encrypted sets is *as fast as any plaintext solution* and determining the set intersection of sets of 100 thousand elements in size can be done in just under a second.

3.2 Preliminaries

A Bloom filter [Blo70] is a data structure that can be used for efficient set membership testing. An (m, k) Bloom filter consists of a bit string bs of length m (indexed using $\mathsf{bs}[\ell]$ for $1 \leq \ell \leq m$) and is associated with k independent hash functions, $h_i \colon \{0,1\}^* \to \{1, \ldots, m\}$ for $1 \leq i \leq k$. The Bloom filter is initialized with the bit string of all zeros. To add an element x to the Bloom filter, we hash the element for each of the k hash functions to obtain $h_i(x)$ and set the $h_i(x)$th position in the bit string bs to 1, *i.e.*, $\mathsf{bs}[h_i(x)] = 1$ for $1 \leq i \leq k$. To test the membership of an element x^*, we simply check if $h_i(x^*) = 1$ for $1 \leq i \leq k$.

Note that Bloom filters have no false negatives for membership testing, but may have false positives. Furthermore, we point out that the hash functions h_i do not necessary need to be cryptographic hash functions.

3.3 Related Work

While the term MC-FE [GGG+14] only recently gained traction, a couple of MC-FE schemes have already been proposed several years ago. For example, for the functionality of summing inputs from distinct clients, Shi *et al.* [SCR+11] proposed a construction. Around the same time, Lewko and Waters [LW11] proposed a multi-authority attribute-based encryption (MA-ABE) scheme. Their construction can also be seen as MC-FE since the evaluated function only outputs a plaintext if the user has the right inputs (*i.e.*, attributes) to the function (*i.e.*, policy). More recently, MC-FE constructions for computing vector equality [KPE+17] and inner products [CDG+18; ABK+19] have been proposed. However, no MC-FE schemes for functionalities related to set operations have been proposed.

Despite being interactive by definition, PSI protocols are functionality-wise the closest related to our constructions. While the concept of PSI dates from the mid-80s [Mea86], renewed interest in PSI protocols started in the beginning of the new millennium [FNP04; KS05]. A comprehensive overview of various PSI constructions and techniques is given by Pinkas, Schneider, and Zohner [PSZ14]. While most PSI constructions achieve their functionality through techniques different from ours, Bloom filters have been used by interactive PSI protocols before [Ker12b; DCW13].

The type of PSI protocols that are most related to our MC-FE schemes are termed *outsourced* PSI [Ker12a; Ker12b; KMR+14; LNZ+14; ATD15; ZX15; ATD16]. In outsourced PSI, a client may upload its encrypted set to a *service provider*, which will then engage in a PSI protocol on the client's behalf. Hence, in outsourced PSI the other client still learns the outcome of the evaluated set intersection, while in our definition of MC-FE for set intersection we require a dedicated evaluator to learn this outcome. This difference is typified by the difference in homomorphic encryption and functional encryption (FE): While both techniques allow us to compute over encrypted data, with homomorphic encryption we learn the *encrypted* output of the computation while with FE we learn the *plaintext* result. The two-client set intersection protocol by Kerschbaum [Ker12a] is a notable exception to regular outsourced PSI: In that construction the service provider also learns the outcome of the set intersection. However, besides their limited scope of considering only two-client set intersection, they consider a weaker security notion. Their

construction is only collusion resistant if the two clients collude against the evaluator, not if the evaluator colludes with one client against the other client (something we show impossible in Section 3.5.1). As a consequence, their construction cannot be extended to a secure scheme in the multi-client case. Moreover, their proposed construction is malleable and thus does not provide any form of integrity.

3.4 Multi-client Functional Encryption for Set Operations

An MC-FE [GGG+14] scheme for a specific set operation consists of n parties, termed *clients*. Each of these clients encrypts their own set. Another party, which we term *evaluator*, having a decryption key and receiving these encrypted sets, can evaluate an n-ary set operation f over the clients' inputs.

To run the same functionality f multiple times without the possibility for the evaluator to mix old clients' inputs with newly received inputs, MC-FE schemes associate an identifier ID with every ciphertext. An evaluator is only able to evaluate the function if all ciphertexts use the same identifier ID.

The MC-FE schemes we propose support only a single functionality f (*e.g.*, set intersection). Therefore, our schemes do not need to define a key generation algorithm to create a decryption key for each of the functionalities. Instead, we can suffice with the creation of a decryption key for the single functionality in Setup. This type of FE schemes is commonly referred to as *single key* [KLM+18]. However, to avoid confusion in our multi-client case—where we still have a key for each client—we refer to this setting as *single evaluation key* MC-FE.

Definition 7 (Multi-client Functional Encryption for Set Operations)**.** A single evaluation key MC-FE scheme for set operation f, consists of the following three polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, n) \to (\mathsf{pp}, \mathsf{esk}, \mathsf{usk}_1, \dots, \mathsf{usk}_n)$**.** On input of the security parameter λ and the number of clients, the algorithm outputs the public parameters pp, the evaluator's evaluation key esk, and the clients' secret keys usk_i for each client $1 \leq i \leq n$. The public parameters are implicitly used in the other algorithms.

$\mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** For a client i to encrypt a set $\mathcal{S}_i$ for identifier ID, the client uses its secret key usk_i and outputs the ciphertext $\mathsf{ct}_{\mathsf{ID},i}$.

$\mathsf{Eval}(\mathsf{esk}, \mathsf{ct}_{\mathsf{ID},1}, \dots, \mathsf{ct}_{\mathsf{ID},n}) \to f(\mathcal{S}_1, \dots, \mathcal{S}_n)$**.** An evaluator having the evaluation key esk and a ciphertext for identifier ID from every client, outputs the function evaluation $f(\mathcal{S}_1, \dots, \mathcal{S}_n)$.

3.4.1 Schemes Without an Evaluator Key

While having schemes with an evaluation secret key might be desirable in some cases, in other cases it is desirable that anyone may learn the outcome of the function, *e.g.*, similar to property-revealing encryption [PR12; BLR+15]. However, observe that we can *always* adapt an MC-FE scheme without an evaluation key to the above defined single evaluation key MC-FE by using public key encryption. Indeed, instead of sending the ciphertexts resulting from the MC-FE scheme directly to the evaluator, we simply require the clients to encrypt these ciphertexts again, but now using the public key of the evaluator. This ensures that only the evaluator with the corresponding private key (used as an evaluation key) can evaluate the functionality f. An alternative solution is to require the clients to send their ciphertexts over a secure channel to the evaluator. This way, no other party has access to the ciphertexts.

We conclude that, since schemes without an evaluation key can be turned into a single evaluation key MC-FE scheme, MC-FE schemes without an evaluation key are at least as powerful as single evaluation key MC-FE. For this reason, *we construct only MC-FE schemes without an evaluation key* and stress that our resulting schemes can thus be used both *with* and *without* an evaluation key.

3.5 Security

We use the indistinguishability-based security notion from Goldwasser *et al.* [GGG+14, § 3.2] for MC-FE. In this notion, the adversary's goal is to decide which of the two, by the adversary chosen, plaintexts is encrypted. The notion allows the adversary to adaptively query for the encryption of plaintext, while it can locally evaluate the received ciphertext using $\mathsf{Eval}(\mathsf{ct}_1, \dots, \mathsf{ct}_n)$. Additionally, the adversary is allowed to statically corrupt the clients by announcing the corrupted clients before it receives the public parameters.

The adversary can thus be seen as a *malicious evaluator* that tries to learn information about the ciphertexts, other than what it should be allowed according to the functionality of the scheme. In its attempts, the malicious evaluator may collude with the clients in an effort to learn more about other clients' ciphertexts.

The security game for our MC-FE schemes without an evaluation key is almost identical to Definition 6 in Chapter 2. The sole difference of the security game for a scheme without an evaluation key is that the adversary cannot query for other evaluation keys (because there are none). Note that, since we are operating on sets, we require the adversary to send two equally sized sets $\mathcal{S}^*_{i,0}, \mathcal{S}^*_{i,1}$, *i.e.*, $|\mathcal{S}^*_{i,0}| = |\mathcal{S}^*_{i,1}|$, for every uncorrupted client $i \in I$.

Note that by using this definition, the ciphertext does not need to hide the set size. This is similar to the semantic security notion where the ciphertext does not need to hide the plaintext size. If this is undesirable, fixed-sized sets can easily be obtained by adding dummy elements to each set.

3.5.1 Corruptions in Two-Client Functional Encryption

We observe that *any* single evaluation key 2C-FE scheme can never be secure against corruptions for non-trivial functionalities. To see why this is the case, consider a 2C-FE scheme for the functionality $f(x, y)$. Assume, without loss of generality, that the adversary corrupts the client which determines the input y. By definition of the game for adaptive IND-security of MC-FE, the adversary submits two values x_0 and x_1 to the challenger. For the challenge inputs to be allowed, it is required that $f(x_0, \cdot) = f(x_1, \cdot)$, *i.e.*, we require $f_{x_0}(y) = f_{x_1}(y)$ *for all possible* y. So, unless f is a constant function in y, we have to require that $x_0 = x_1$, for which it is trivial to see that the challenge will be indistinguishable.

Generalizing the result, we see that in an MC-FE scheme for n clients, at least two clients need to remain uncorrupted. Phrased differently, this means that for MC-FE with n clients, we can allow for at most $n - 2$ corruptions.

The corruption of a client can also be seen as appointing a client as the evaluator. This means that in a two-client case, none of the clients can be the evaluator, as this would always lead to an insecure scheme since the evaluator is able to determine the other client's input by altering its own input.

3.6 Two-Client Constructions for Set Intersections

We propose several 2C-FE schemes for various set operations: computing the cardinality of the set intersection, computing the set intersection itself, computing the set intersection with data transfer or projection, and computing the set intersection only if a threshold is reached. We discuss constructions supporting more than two clients in Section 3.7.

3.6.1 Two-Client Set Intersection Cardinality

To compute the cardinality of a set intersection from two clients, we can suffice with a simple scheme using a pseudorandom function (PRF) (see Definition 1). The two clients encrypt each set element individually using a PRF under the same key. Since a PRF has a deterministic output, the evaluator can now use *any* algorithm for determining the cardinality of the intersection,

even algorithms that only operate on plaintext data (*e.g.*, see [DK11] for an overview).

Setup$(1^\lambda) \to (\mathsf{pp}, \mathsf{usk}_1, \mathsf{usk}_2)$**.** Let $\Phi = \{\phi_\kappa\}$ be a PRF ensemble for functions $\phi_\kappa \colon \mathcal{ID} \times \{0,1\}^* \to \{0,1\}^{\geq\lambda}$. Pick a PRF ϕ_{msk}. The public parameters are $\mathsf{pp} = (\Phi)$ and the clients' keys $\mathsf{usk}_1 = \mathsf{usk}_2 = (\phi_{\mathsf{msk}})$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** For a client i to encrypt its set $\mathcal{S}_i$ for an identifier $\mathsf{ID} \in \mathcal{ID}$, the client computes the PRF for each set element $x_j \in \mathcal{S}_i$. It outputs the set $\mathsf{ct}_{\mathsf{ID},i} = \{\, \phi_{\mathsf{msk}}(\mathsf{ID}, x_j) \mid x_j \in \mathcal{S}_i \,\}$.

Eval$(\mathsf{ct}_{\mathsf{ID},1}, \mathsf{ct}_{\mathsf{ID},2}) \to |\mathcal{S}_1 \cap \mathcal{S}_2|$**.** To evaluate the cardinality of the set intersection, output $\left|\mathsf{ct}_{\mathsf{ID},1} \cap \mathsf{ct}_{\mathsf{ID},2}\right|$.

We can use a block cipher, keyed-hash function, hash-based message authentication code, or a similar function as the PRF.

Theorem 1. *The two-client set intersection cardinality scheme defined above is secure under the assumption that the* PRF *is indistinguishable from a random function.*

Proof. This directly follows from the security of the PRF. Note that the evaluator only learns whether two set elements $x_{1,j} \in \mathcal{S}_1$ and $x_{2,j'} \in \mathcal{S}_2$ are equal or not. Nothing else is revealed about the set elements $x_{1,j}$ and $x_{2,j'}$. □

3.6.2 Two-Client Set Intersection

In case of two-client set intersection, we need not only to determine whether two encrypted set elements are the same, but also learn the plaintext set element if they are the same. We achieve this by adapting our construction for two-client set intersection cardinality with a combination of convergent encryption [DAB+02] (*cf.* message-locked encryption [BKR13]) and secret sharing: We encrypt the set element under a key derived from the message itself and secret share the encryption key. If both clients encrypted the same message, the decryption key can be recovered from the secret shares and the ciphertext can be decrypted. To encrypt the set element itself, we use an authenticated encryption (AE) scheme [BN00].

Setup$(1^\lambda) \to (\mathsf{pp}, \mathsf{usk}_1, \mathsf{usk}_2)$**.** Let $\langle g \rangle = \mathbb{G}$ be a group of prime order p and let $\Phi = \{\phi_\kappa\}$ be a PRF ensemble for functions $\phi_\kappa \colon \mathcal{ID} \times \{0,1\}^* \to \mathbb{G}$ and AE an AE scheme. Define a mapping from the group to the key space of the AE scheme, $H \colon \mathbb{G} \to \mathcal{K}_{\mathsf{AE}}$. Pick a PRF ϕ_{msk} and pick $\sigma_1 \xleftarrow{R} \mathbb{Z}_p$ to set $\sigma_2 = 1 - \sigma_1 \pmod p$. The public parameters are $\mathsf{pp} = (\mathbb{G}, \Phi, H, \mathsf{AE})$ and the clients' keys $\mathsf{usk}_1 = (\phi_{\mathsf{msk}}, \sigma_1)$ and $\mathsf{usk}_2 = (\phi_{\mathsf{msk}}, \sigma_2)$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** For a client i to encrypt its set $\mathcal{S}_i$ for an identifier $\mathsf{ID} \in \mathcal{ID}$, the client computes the PRF for each set element $x_j \in \mathcal{S}_i$. It outputs the set of tuples $\{(\mathsf{ct}_{\mathsf{ID},i,j,1}, \mathsf{ct}_{\mathsf{ID},i,j,2})\}_{1 \le j \le |\mathcal{S}_i|}$,

$$\mathsf{ct}_{\mathsf{ID},i} = \left\{ \left(k_{\mathsf{ID},j}^{\sigma_i}, \mathsf{AE.Enc}_{H(k_{\mathsf{ID},j})}(x_j)\right) \mid k_{\mathsf{ID},j} = \phi_{\mathsf{msk}}(\mathsf{ID}, x_j), x_j \in \mathcal{S}_i \right\}.$$

Eval$(\mathsf{ct}_{\mathsf{ID},1}, \mathsf{ct}_{\mathsf{ID},2}) \to \mathcal{S}_1 \cap \mathcal{S}_2$**.** For all $\mathsf{ct}_{\mathsf{ID},1,j,2} = \mathsf{ct}_{\mathsf{ID},2,k,2}$ (and hence $x = x_{1,j} = x_{2,k}$), determine

$$\begin{aligned} k_{\mathsf{ID},x} &= \mathsf{ct}_{\mathsf{ID},1,j,1} \cdot \mathsf{ct}_{\mathsf{ID},2,k,1} \\ &= \phi_{\mathsf{msk}}(\mathsf{ID}, x)^{\sigma_1} \cdot \phi_{\mathsf{msk}}(\mathsf{ID}, x)^{\sigma_2} \\ &= \phi_{\mathsf{msk}}(\mathsf{ID}, x)^{\sigma_1 + \sigma_2} \\ &= \phi_{\mathsf{msk}}(\mathsf{ID}, x), \end{aligned}$$

to decrypt $\mathsf{ct}_{\mathsf{ID},i,j,2}$ using $\mathsf{AE.Dec}_{H(k_{\mathsf{ID},x})}(\mathsf{ct}_{\mathsf{ID},i,j,2})$ for $i = 1$ or for $i = 2$.

Theorem 2. *The two-client set intersection scheme defined above is secure under the decisional Diffie–Hellman (DDH) assumption, a secure PRF, and a secure AE scheme.*

Proof. We construct an algorithm that is able to break the DDH problem if a probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ has a non-negligible advantage in winning the game.

Setup The challenger $\mathcal{B}$ receives the DDH tuple (g, g^a, g^b, T) from the group $\mathbb{G}$ of prime order p. It defines a PRF ensemble $\Phi = \{\phi_\kappa\}$ and mapping $H\colon \mathbb{G} \to \mathcal{K}_{\mathsf{AE}}$ according to the scheme. The public parameters $\mathsf{pp} = (\mathbb{G}, \Phi, H, \mathsf{AE})$ are sent to the adversary. The challenger indirectly sets $\sigma_1 = a$ and $\sigma_2 = 1 - a$, *i.e.*, $g^{\sigma_1} = g^a$ and $g^{\sigma_2} = g \cdot (g^a)^{-1}$.

Query Upon receiving an allowed encryption query for $(i, \mathsf{ID}, \mathcal{S})$, the challenger encrypts the elements of the set $\mathcal{S}$ as follows. It models the PRF as follows: On input (ID, x_j), output $g^{r_{\mathsf{ID},x_j}}$, where, if the input has not been queried before, $r_{\mathsf{ID},x_j} \xleftarrow{R} \mathbb{Z}_p$. The challenger encrypts an element $x_j \in \mathcal{S}$ as

$$\mathsf{ct}_{\mathsf{ID},i,j} = \begin{cases} \left((g^a)^{r_{\mathsf{ID},x_j}}, \mathsf{AE.Enc}_k(x_j)\right) & \text{if } i = 1; \\ \left((g \cdot (g^a)^{-1})^{r_{\mathsf{ID},x_j}}, \mathsf{AE.Enc}_k(x_j)\right) & \text{if } i = 2, \end{cases}$$

where $k = H(g^{r_{\mathsf{ID},x_j}})$. It outputs the encrypted set $\mathsf{ct}_{\mathsf{ID},i}$ to the adversary.

Challenge An allowed challenge request from the adversary for the sets $\mathcal{S}^*_{1,0}$, $\mathcal{S}^*_{1,1}$, $\mathcal{S}^*_{2,0}$, and $\mathcal{S}^*_{2,1}$ with identifier ID^*, is answered by the challenger by

sending the encrypted sets $\mathcal{S}^*_{1,b}$ and $\mathcal{S}^*_{2,b}$ back to the adversary. An element $x_j \notin (\mathcal{S}_{1,b} \cap \mathcal{S}_{2,b})$ is encrypted as

$$\mathsf{ct}_{\mathsf{ID},i,j} = \begin{cases} (T^{r_{\mathsf{ID}^*,x_j}}, \mathsf{AE.Enc}_k(x_j)) & \text{if } i = 1; \\ ((g^b \cdot T^{-1})^{r_{\mathsf{ID}^*,x_j}}, \mathsf{AE.Enc}_k(x_j)) & \text{if } i = 2, \end{cases}$$

where $k = H((g^b)^{r_{\mathsf{ID},x_j}})$. Note that this indirectly sets the output of the PRF to $g^{br_{\mathsf{ID}^*,x_j}}$ for $x_j \notin (\mathcal{S}_{1,b} \cap \mathcal{S}_{2,b})$. The elements $x_j \in (\mathcal{S}_{1,b} \cap \mathcal{S}_{2,b})$ are encrypted as in the query phase.

If the adversary $\mathcal{A}$ outputs a correct guess $b' = b$, the challenger outputs the guess that $T = g^{ab}$, otherwise, it outputs its guess $T \in_R \mathbb{G}$. □

3.6.3 Two-Client Set Intersection with Data Transfer or Projection

The two-client set intersection scheme described above can be extended into a two-client set intersection scheme with data transfer (analogous to PSI *with data transfer* [DT10; JL10]). Instead of only encrypting the set element x_j itself, $\mathsf{ct}_{\mathsf{ID},i,j,2} = \mathsf{AE.Enc}_k(x_j)$, we can also choose to encrypt both the element itself and the data associated to the set element $\rho(x_j)$. The security of the scheme is the same as before since we rely on the security of the AE scheme.

Moreover, the proposed scheme also allows for a two-client set intersection projection scheme (analogous to PSI *with projection* [CFS+17]). We construct such a scheme by encrypting only the associated data $\rho(x_j)$, $\mathsf{ct}_{\mathsf{ID},i,j,2} = \mathsf{AE.Enc}_k(\rho(x_j))$, not the set element x_j itself. Security follows from the fact that the AE decryption key $k = H(\phi_{\mathsf{msk}}(\mathsf{ID}, x_j))$ does not reveal any information about the set element x_j, assuming the security of the used PRF. However, the evaluator does learn that the projections of both clients correspond to the same set element.

Lastly, we can also extend the scheme by encrypting the ciphertexts of any other MC-FE scheme to achieve even more advanced functionality. For example, we could encrypt the ciphertext of an MC-FE scheme for summing [SCR+11] as the associated data. With such a scheme, we directly achieve a non-interactive variant of a recently proposed PSI protocol [IKN+19].

3.6.4 Two-Client Threshold Set Intersection

To allow the evaluator to learn the cardinality of the intersection, but only the set elements in the intersection if the clients have at least t set elements in common, we propose a two-client threshold set intersection scheme. We achieve this by encrypting the share of the decryption key for the AE

ciphertext $k_{\mathsf{ID},j}^{\sigma_i}$ using another encryption key. This newly added encryption key can only be obtained by the evaluator if the clients have at least t set elements in common.

Although the construction is based on the previous scheme, the precise construction is quite technical. We therefore state the complete scheme below.

Setup$(1^\lambda, t) \to (\mathsf{pp}, \mathsf{usk}_1, \mathsf{usk}_2)$**.** Let AE be an AE scheme and $\langle g \rangle = \mathbb{G}$ be a group and $\mathbb{F}_p$ be a field, both of prime order p. Let $\Phi = \{\phi_\kappa\}$ and $\Psi = \{\psi_\kappa\}$ be PRF ensembles for functions $\phi_\kappa \colon \mathcal{ID} \times \{0,1\}^* \to \mathbb{G}$ and $\psi_\kappa \colon \mathcal{ID} \times \{0,1\}^* \to \mathbb{F}_p$, respectively. Define a mapping from the group to the key space of the AE scheme, $H \colon \mathbb{G} \to \mathcal{K}_{\mathsf{AE}}$. Pick three PRFs $\phi \in \Phi, \psi_1, \psi_2 \in \Psi$ and $\sigma_1 \xleftarrow{R} \mathbb{Z}_p, \rho_1 \xleftarrow{R} \mathbb{Z}_{p-1}$, setting $\sigma_2 = 1 - \sigma_1 \pmod p$ and $\rho_2 = 1 - \rho_1 \pmod{p-1}$.

The public parameters are $\mathsf{pp} = (\mathbb{G}, \Phi, \Psi, H, \mathsf{AE}, t)$ and the clients' secret keys $\mathsf{usk}_1 = (\phi, \psi_1, \psi_2, \sigma_1, \rho_1)$ and $\mathsf{usk}_2 = (\phi, \psi_1, \psi_2, \sigma_2, \rho_2)$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** For a client i to encrypt its set $\mathcal{S}_i$ for an identifier $\mathsf{ID} \in \mathcal{ID}$, the client computes the PRF for each set element $x_j \in \mathcal{S}_i$. It defines the $(t-1)$th degree polynomial f_{ID} by setting the coefficients $c_i = \psi_2(\mathsf{ID}, i)$, for $0 \le i < t$, to obtain the polynomial $f_{\mathsf{ID}}(x) = c_{t-1}x^{t-1} + \cdots + c_1 x + c_0$.

The client outputs the set

$$\mathsf{ct}_{\mathsf{ID},i} = \Big\{ (k_{\mathsf{ID},j,2}, f(k_{\mathsf{ID},j,2})^{\rho_i}, \mathsf{AE.Enc}_{H(c_0)}(k_{\mathsf{ID},j,1}^{\sigma_i}), \mathsf{AE.Enc}_{H(k_{\mathsf{ID},j,1})}(x_j)) \\ \mid k_{\mathsf{ID},j,1} = \phi(\mathsf{ID}, x_j), k_{\mathsf{ID},j,2} = \psi_1(\mathsf{ID}, x_j), x_j \in \mathcal{S}_i \Big\}.$$

Eval$(\mathsf{ct}_{\mathsf{ID},1}, \mathsf{ct}_{\mathsf{ID},2}) \to (|\mathcal{S}_1 \cap \mathcal{S}_2|, \{ x_j \mid x_j \in \mathcal{S}_1 \cap \mathcal{S}_2, |\mathcal{S}_1 \cap \mathcal{S}_2| \ge t \})$**.** The evaluation algorithm consists of two stages; the second stage is only executed if $|\mathcal{S}_1 \cap \mathcal{S}_2| \ge t$.

1. To determine the cardinality of the set intersection $|\mathcal{S}_1 \cap \mathcal{S}_2|$, the evaluator counts the number of times a value $k_{\mathsf{ID},j,2}$ occurs both in $\mathsf{ct}_{\mathsf{ID},1}$ and $\mathsf{ct}_{\mathsf{ID},2}$.

2. If $|\mathcal{S}_1 \cap \mathcal{S}_2| \ge t$, the evaluator uses Lagrange interpolation to compute the value $c_0 = f(0)$. It can do so by taking t distinct tuples $(k_{\mathsf{ID},j,2}, f(k_{\mathsf{ID},j,2}))$, where $f(k_{\mathsf{ID},j,2}) = f(k_{\mathsf{ID},j,2})^{\rho_1} \cdot f(k_{\mathsf{ID},j,2})^{\rho_2}$. Now, when the secret c_0 has been recovered from the shares, the evaluator can use it to decrypt the values $\mathsf{AE.Enc}_{H(c_0)}(k_{\mathsf{ID},j,1}^{\sigma_i})$. So, the evaluator obtains $k_{\mathsf{ID},j,1}^{\sigma_i}$ for *every* set element in $x_j \in \mathcal{S}_i$ if $|\mathcal{S}_1 \cap \mathcal{S}_2| \ge t$. Observe that for the elements in the intersection, the evaluator has

both $k_{\mathsf{ID},j,1}^{\sigma_1}$ and $k_{\mathsf{ID},j,1}^{\sigma_2}$, and can compute $k_{\mathsf{ID},j,1} = k_{\mathsf{ID},j,1}^{\sigma_1} \cdot k_{\mathsf{ID},j,1}^{\sigma_2}$. Finally, using $H(k_{\mathsf{ID},j,1})$, it can decrypt $\mathsf{AE.Enc}_{H(k_{\mathsf{ID},j,1})}(x_j)$ to obtain $x_j \in \mathcal{S}_1 \cap \mathcal{S}_2$.

Since the construction above builds upon the set intersection scheme, which can be modified into a set intersection with data transfer scheme or a set intersection with projection scheme, we similarly obtain both threshold set intersection with data transfer and projection.

Theorem 3. *The two-client threshold set intersection scheme defined above is secure under the* DDH *assumption, a secure* PRF*, and a secure* AE *scheme.*

Proof. We only have to prove that the values $k_{\mathsf{ID},j,1}^{\sigma_i}$ can only be obtained if $|\mathcal{S}_1 \cap \mathcal{S}_2| \geq t$, as the rest of the proof directly follows from Theorem 2. Since the values $k_{\mathsf{ID},j,1}^{\sigma_i}$ are encrypted using an AE scheme using the key $H(c_0)$, the values are only know to the evaluator if it has the key $H(c_0)$ (under the assumption of a secure AE scheme). The fact that c_0 (and hence $H(c_0)$) can only be obtained from the secret shares follows from the information-theoretic security of Shamir's secret sharing scheme if a random polynomial f_{ID} was used. Note that the $(t-1)$th degree polynomial is random under the assumption of a secure PRF. Finally, using a similar argument as in Theorem 2, we can show that, under the DDH assumption, $f(k_{\mathsf{ID},j,2})^{\rho_1}$ or $f(k_{\mathsf{ID},j,2})^{\rho_2}$ does not reveal any information about $f(k_{\mathsf{ID},j,2})$ if $f(k_{\mathsf{ID},j,2})^{\rho_2}$ or $f(k_{\mathsf{ID},j,2})^{\rho_1}$, respectively, is unknown. □

3.7 Multi-client Constructions for Set Intersections

While the 2C-FE constructions from Section 3.6 could be used in a multi-client case, this would leak information about each pair of sets. For the same reason, deterministic encryption cannot be used in secure MC-FE constructions, which makes it much harder to develop efficient MC-FE schemes.

3.7.1 Multi-client Set Intersection Cardinality

We construct an MC-FE scheme for testing the set intersection using only a hash function and secret sharing. The proposed scheme incurs no additional leakage and is proven adaptive IND-secure. While our scheme has an evaluation algorithm which does not rely on heavy cryptographic machinery and runs in polynomial time (for a fixed number of clients n), it is not very efficient. The running time of the evaluation algorithm grows in the product of the cardinality of the individual clients' set size. However, for relatively

small sets or a small number of clients this scheme might still be efficient enough to use in practice.

Setup$(1^\lambda, n) \to (\mathsf{pp}, \mathsf{usk}_1, \dots, \mathsf{usk}_n)$**.** Let $\langle g \rangle = \mathbb{G}$ be a group of prime order p and let $H\colon \mathcal{ID} \times \{0,1\}^* \to \mathbb{G}$ be a hash function. Create random shares of 0 by picking $\sigma_i \xleftarrow{R} \mathbb{Z}_p$, for all $2 \leq i \leq n$, and setting $\sigma_1 = -\sum_{i=2}^{n} \sigma_i \pmod p$. The public parameters are $\mathsf{pp} = (H)$ and the clients' keys $\mathsf{usk}_i = (\sigma_i)$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** For a client i to encrypt its set $\mathcal{S}_i$ using an identifier $\mathsf{ID} \in \mathcal{ID}$, the client encrypts each set element $x_j \in \mathcal{S}_i$ individually. It outputs the set $\mathsf{ct}_{\mathsf{ID},i} = \left\{ H(\mathsf{ID}, x_j)^{\sigma_i} \mid x_j \in \mathcal{S}_i \right\}$.

Eval$(\mathsf{ct}_{\mathsf{ID},1}, \dots, \mathsf{ct}_{\mathsf{ID},n}) \to \left|\bigcap_{i=1}^{n} \mathcal{S}_i\right|$**.** For each n-tuple $(c_1, \dots, c_n) \in \mathsf{ct}_{\mathsf{ID},1} \times \cdots \times \mathsf{ct}_{\mathsf{ID},n}$, the evaluator evaluates $\prod_{i=1}^{n} c_i \stackrel{?}{=} 1$. The evaluator outputs the count for the number of times the expression above evaluates to TRUE.

We prove the construction secure under selective corruptions, but we note that it is also possible to achieve a proof under dynamic corruptions (although less tight) by adapting the proofs from Shi *et al.* [SCR+11].

Theorem 4. *The improved multi-client set intersection cardinality scheme defined above is secure up to* $(n-2)$ *corruptions under the* DDH *assumption in the random oracle model* (ROM).

Proof. Let $\mathcal{A}$ be a p.p.t. adversary playing the adaptive IND-security game for MC-FE. We show how to use $\mathcal{A}$ as a distinguisher for a DDH tuple, winning with a non-negligible advantage if $\mathcal{A}$ has a non-negligible advantage in winning the security game.

Random Oracle On input of a tuple (ID, x_j) the oracle checks if it has answered the query before. If not, it picks a value $\beta_{\mathsf{ID},x_j} \xleftarrow{R} \mathbb{Z}_p$. Next, the challenger $\mathcal{B}$ guesses whether the query is for the challenge ID. If so, the oracle outputs $(g^b)^{\beta_{\mathsf{ID},x_j}}$, otherwise, it outputs $g^{\beta_{\mathsf{ID},x_j}}$. If the guess turns out to be wrong later, $\mathcal{B}$ can simply abort the game.

Corruptions The adversary $\mathcal{A}$ announces the set of uncorrupted and corrupted clients, I and $\bar{I}$, respectively.

Setup For $i \in \bar{I}$, the challenger $\mathcal{B}$ picks $\sigma_i \xleftarrow{R} \mathbb{Z}_p$ and sends the values to the adversary $\mathcal{A}$. Let $i' \in I$, for $i \in I \setminus \{i'\}$, $\mathcal{B}$ indirectly sets $\sigma_i = a \cdot \alpha_i$, where $\alpha_i \xleftarrow{R} \mathbb{Z}_p$, by setting $g^{\sigma_i} = (g^a)^{\alpha_i}$. For i', it indirectly sets $\sigma_{i'} = -\sum_{i \neq i'} \sigma_i$,

$$g^{\sigma_{i'}} = \prod_{i \in \bar{I}} g^{-\sigma_i} \cdot \prod_{i \in I, i \neq i'} (g^a)^{-\alpha_i}.$$

Query To answer an encryption query $\mathcal{S}_i$ for an uncorrupted client $i \in I$, the challenger uses the oracle to obtain $\{ \beta_{\mathsf{ID},x_j} \mid x_j \in \mathcal{S}_i \}$ and construct the ciphertext as $\mathsf{ct}_{\mathsf{ID},i} = \{ (g^{\sigma_i})^{\beta_{\mathsf{ID},x_j}} \mid x_j \in \mathcal{S}_i \}$.

Challenge Upon receiving the challenge sets $\{ (\mathcal{S}^*_{i,0}, \mathcal{S}^*_{i,1}) \mid i \in I \}$ and an ID^* from the adversary, the challenger picks $b \xleftarrow{R} \{0,1\}$. The challenger returns the ciphertexts

$$\mathsf{ct}_{\mathsf{ID}^*,i'} = \left\{ \prod_{i \in \bar{I}} (g^b)^{-\sigma_i \cdot \beta_{\mathsf{ID}^*,x_j}} \cdot \prod_{i \in I, i \neq i'} T^{-\alpha_i \cdot \beta_{\mathsf{ID}^*,x_j}} \mid x_j \in \mathcal{S}^*_{i,b} \right\} \qquad \text{and}$$

$$\mathsf{ct}_{\mathsf{ID}^*,i} = \left\{ T^{\alpha_i \beta_{\mathsf{ID}^*,x_j}} \mid x_j \in \mathcal{S}^*_{i,b} \right\} \qquad \text{for } i \neq i'.$$

Note that if $T = g^{ab}$, the ciphertext is distributed properly according the scheme. If $T \in_R \mathbb{G}$, the challenger returns a ciphertext of a randomly distributed set element. So, the challenger $\mathcal{B}$ guesses that $T = g^{ab}$ if $\mathcal{A}$ correctly guessed $b' = b$ and otherwise, $\mathcal{B}$ guesses that $T \in_R \mathbb{G}$. □

We remark that while the security of the two-client schemes could be proven in the standard model, our multi-client constructions can only be proven in the ROM. The difference in the constructions is that in the two-client case, no corruptions are taken place, and thus we can use a programmable PRF instead of a programmable random oracle.

3.7.2 Efficient Multi-client Set Intersection Cardinality

A drawback of the multi-client set intersection cardinality scheme might be that the computational complexity for the evaluator grows quickly in the total number of set elements (*i.e.*, $\prod_{i=1}^{n} |\mathcal{S}_i|$). To address this problem, we propose an alternative scheme using Bloom filters. In this scheme, we first combine the Bloom filter representation of every client's set in the encrypted domain, resulting in an encrypted Bloom filter representing the intersection of all clients' sets. Next, the evaluator uses the encrypted set elements of *any* client to determine the cardinality of the intersection. This method used by the evaluator to determine the cardinality of the intersection can be seen as computing $\left|\mathcal{S}_i \cap \left(\bigcap_{i=1}^{n} \mathcal{S}_i\right)\right| = \left|\bigcap_{i=1}^{n} \mathcal{S}_i\right|$. The theoretical efficiency of $\mathcal{O}(n + \min_{i=1}^{n} |\mathcal{S}_i|)$ ciphertext operations is much better than the other scheme. However, the proposed scheme is only secure if no corruptions are taking place.

Setup$(1^\lambda, n, m, k) \rightarrow (\mathsf{pp}, \mathsf{usk}_1, \ldots, \mathsf{usk}_n)$**.** Let $\langle g \rangle = \mathbb{G}$ be a group of prime order p and let BF be a specification for an (m,k) Bloom filter. Let $\Phi = \{\phi_\kappa\}$ be a PRF ensemble for functions $\phi_\kappa \colon \{0,1\}^* \rightarrow \{0,1\}^{\geq \lambda}$ and let $H \colon \mathcal{ID} \times$

$\{0,1\}^* \to \mathbb{G}$ be a hash function. Pick a PRF $\phi \in \Phi$. Additionally, pick for $1 \leq i \leq n$, values $c_i \xleftarrow{R} \mathbb{Z}_p$ and define the n-degree polynomial $f(x) = c_n x^n + \cdots + c_1 x$ over the field $\mathbb{F}_p$. The public parameters are $\mathsf{pp} = (\mathsf{BF}, \Phi, H)$ and the clients' secret keys are $\mathsf{usk}_i = (\phi, f(i), f(n+i))$ for $1 \leq i \leq n$. Note that every client receives the same PRF ϕ, but different secret shares $f(i)$ and $f(n+i)$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to (\mathsf{ct}_{\mathsf{ID},i,\mathsf{bs}_\mathcal{S}}, \mathsf{ct}_{\mathsf{ID},i,\mathcal{S}})$**.** First, the client initializes the Bloom filter to obtain $\mathsf{bs}_\mathcal{S}$. Next, it adds its encrypted set elements, $\phi(x_j)$ for $x_j \in \mathcal{S}_i$, to the Bloom filter. For each $1 \leq \ell \leq m$, the client sets $r_{i,\ell} \xleftarrow{R} \mathbb{Z}_p$, if $\mathsf{bs}_\mathcal{S}[\ell] = 0$, and $r_{i,\ell} = 0$, otherwise. The client encrypts the Bloom filter for $\mathsf{bs}_\mathcal{S}$ as the ordered set

$$\mathsf{ct}_{\mathsf{bs}_\mathcal{S}} = \left\{ H(\mathsf{ID}, \ell)^{f(i)} \cdot g^{r_{i,\ell}} \mid 1 \leq \ell \leq m \right\}.$$

Additionally, the client initializes a new bit string bs_j for each element $x_j \in \mathcal{S}_i$. It encrypts each element x_j and adds $\phi(x_j)$ to the Bloom filter for bs_j. Let t_j denote the Hamming weight (*i.e.*, the number of 1s) of the resulting bit string bs_j. For the resulting bit string bs_j pick $r_{i,j,\ell} \xleftarrow{R} \mathbb{Z}_p$ for $1 \leq \ell \leq m$. Additionally, it sets $\rho_{i,j,\ell} \xleftarrow{R} \mathbb{Z}_p$ if $\mathsf{bs}_j[\ell] = 0$, and $\rho_{i,j,\ell} = t_j \cdot r_{i,j,\ell}$ otherwise. It encrypts the Bloom filter for bs_j as

$$\mathsf{ct}_{\mathsf{bs}_j} = \left(\left\{ H(\mathsf{ID}, \ell)^{f(n+i)} \cdot g^{\rho_{i,j,\ell}}, g^{r_{i,j,\ell}} \mid 1 \leq \ell \leq m \right\} \right).$$

Finally, the client outputs the ciphertext $\left(\mathsf{ct}_{\mathsf{bs}_\mathcal{S}}, \left\{ \mathsf{ct}_{\mathsf{bs}_j} \mid x_j \in \mathcal{S}_i \right\} \right)$.

Eval$(\mathsf{ct}_{\mathsf{ID},1}, \ldots, \mathsf{ct}_{\mathsf{ID},n}) \to \left| \bigcap_{i=1}^n \mathcal{S}_i \right|$**.** Since the clients' ciphertext are encryptions of the individual set elements, we can determine a client with the smallest (encrypted) set. Let γ be such a client. Now, for $1 \leq \ell \leq m$, compute the partial Lagrange interpolation

$$a_\ell = \prod_{i=1}^{n} \left(\mathsf{ct}_{\mathsf{ID},i,\mathsf{bs}_\mathcal{S}[\ell]} \right)^{\Delta_{\{1,\ldots,n,n+\gamma\},i}}.$$

Set $d = 0$. Next, to determine if an encrypted set element $x_j \in \mathcal{S}_\gamma$ (represented by a tuple $(\mathsf{ct}_{\mathsf{ID},\gamma,\mathsf{bs}_j}, g^{r_{\gamma,j,\ell}}) \in \mathsf{ct}_{\mathsf{ID},\gamma,\mathcal{S}}$) is in the intersection of all sets, check for each $1 \leq \ell \leq m$, if

$$\left(\mathsf{ct}_{\mathsf{ID},\gamma,\mathsf{bs}_j[\ell]} \right)^{\Delta_{\{1,\ldots,n,n+\gamma\},n+\gamma}} \cdot a_\ell \stackrel{?}{=} (g^{r_{\gamma,j,\ell}})^{t_{j,\ell} \cdot \Delta_{\{1,\ldots,n,n+\gamma\},n+\gamma}}$$

for values $1 \leq t_{j,\ell} \leq k$. If the value $t_{j,\ell}$ occurs $t_{j,\ell}$ times for the values $1 \leq \ell \leq m$, increase the value d by one.

After all encrypted set element $x_j \in \mathcal{S}_\gamma$ have been checked, output the cardinality of the set intersection d.

Correctness To see that the above defined scheme is correct, observe that if a set element $x_j \in \mathcal{S}_i$ is in the intersection of all clients' sets, the values $r_{i,j,\ell}$ equal 0 for the same values of ℓ in the encrypted Bloom filters $\mathsf{ct}_{\mathsf{ID},i,\mathsf{bs}_\mathcal{S}}$. Hence, by using the Lagrange interpolation on these elements (corresponding to a_ℓ) together with an encrypted Bloom filter for a single set element $x_j \in \mathcal{S}_\gamma$ (corresponding to $\mathsf{ct}_{\mathsf{ID},\gamma,\mathsf{bs}_j}$), we obtain

$$H(\mathsf{ID},\ell)^{f(0)} \cdot g^{r_{i,j,\ell}\cdot\Delta_{\{1,\ldots,n,n+\gamma\},n+\gamma}} = g^{r_{i,j,\ell}\cdot\Delta_{\{1,\ldots,n,n+\gamma\},n+\gamma}}.$$

Now, note that we set $\rho_{i,j,\ell} = t_j \cdot r_{i,j,\ell}$ if the bit string value $\mathsf{bs}_j[\ell] = 1$. So, if exactly t_j bit string values in the set intersection are set to 1, we know that the element is a member of the set intersection.

Theorem 5. *The improved multi-client set intersection cardinality scheme defined above is secure without corruptions under the* DDH *assumption in the* ROM.

Proof. We construct an algorithm that is able to break the DDH problem if a p.p.t. adversary $\mathcal{A}$ has a non-negligible advantage in winning the game.

Random Oracle On input of a tuple (ID,ℓ) the oracle checks if it has answered the query before. If not, it picks a value $\beta_{\mathsf{ID},\ell} \xleftarrow{R} \mathbb{Z}_p$. Next, the challenger $\mathcal{B}$ guesses whether the query is for the challenge ID. If so, the oracle outputs $(g^b)^{\beta_{\mathsf{ID},\ell}}$, otherwise, it outputs $g^{\beta_{\mathsf{ID},\ell}}$. If the guess turns out to be wrong later, $\mathcal{B}$ can simply abort the game.

Setup The challenger $\mathcal{B}$ receives the DDH tuple (g, g^a, g^b, T) from the group $\mathbb{G}$ of prime order p. It defines a PRF ensemble $\Phi = \{\phi_\kappa\}$ and the Bloom filter BF according to the scheme. Pick for $1 \le i \le n$, values $c_i \xleftarrow{R} \mathbb{Z}_p$ and define the n-degree polynomial $f'(x) = c_n x^n + \cdots + c_1 x$ over the field $\mathbb{F}_p$. The challenger uses $f(x) = a \cdot f'(x)$ to indirectly define the secret shares. Note that this still allows $\mathcal{B}$ to compute $g^{f(x)} = (g^a)^{f'(x)}$ for all values of x.

Query To answer an encryption query $\mathcal{S}_i$ for a client i, the challenger uses the oracle to obtain $\{\, \beta_{\mathsf{ID},\ell} \mid x_j \in \mathcal{S}_i \,\}$ and construct the ciphertext as in the scheme, but using $H(\mathsf{ID},\ell)^{f(x)} = (g^a)^{\beta_{\mathsf{ID},\ell} f'(x)}$.

Challenge Upon receiving the challenge sets $(\mathcal{S}^*_{i,0}, \mathcal{S}^*_{i,1})$ for $1 \le i \le n$ and an ID^* from the adversary, the challenger picks $b \xleftarrow{R} \{0,1\}$. The challenger returns the encryptions of the sets $\mathcal{S}^*_{i,b}$ using the scheme's encrypt algorithm, but replacing $H(\mathsf{ID}^*,\ell)^{f(x)}$ by $T^{\beta_{\mathsf{ID}^*,\ell} f'(x)}$. Note that if $T = g^{ab}$, the ciphertext is distributed properly according the scheme. If $T \in_R \mathbb{G}$, the challenger returns a ciphertext of a randomly distributed set element. So, the challenger $\mathcal{B}$ guesses that $T = g^{ab}$ if $\mathcal{A}$ correctly guessed $b' = b$ and otherwise, $\mathcal{B}$ guesses that $T \in_R \mathbb{G}$. □

To construct efficient MC-FE schemes for set operations that resist corruptions, we need to be able to check the membership of an encrypted set element against the encrypted intersection of the clients' sets. The above construction fails to be secure against corruptions as it (partially) reveals the individual bits in the bit string of a Bloom filter for a set element, *i.e.*, the adversary learns (part of) the bit string representation of the set element. In Chapter 6, we present an idea for getting both an efficient MC-FE scheme and have it secure against corruptions.

3.7.3 Multi-client Set Intersection

Set intersections can be computed using a notion similar to non-interactive distributed encryption (DE) schemes [GH11; LHK14]. A DE scheme is characterized by two distinctive features. Firstly, we have that multiple clients can encrypt a plaintext under their own secret key. Secondly, if enough clients have encrypted the same plaintext, anyone can recover this plaintext from the clients' ciphertexts.

We construct an MC-FE scheme for set intersection from a DE scheme.

Setup$(1^\lambda, n) \to (\mathsf{pp}, \mathsf{usk}_1, \dots, \mathsf{usk}_n)$**.** Run $\mathsf{DE.Gen}(1^\lambda, n, n)$ to generate an n-OUT-OF-n DE scheme defined by pp and obtain the encryption keys $(\mathsf{usk}_1, \dots, \mathsf{usk}_n)$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, \mathcal{S}_i) \to \mathsf{ct}_{\mathsf{ID},i}$**.** To encrypt the set $\mathcal{S}_i$, encrypt the identifier ID together with each set element $x_j \in \mathcal{S}_i$ individually,

$$\mathsf{ct}_{\mathsf{ID},i} = \left\{ \mathsf{DE.Enc}(\mathsf{usk}_i, \mathsf{ID} \,\|\, x_j) \mid x_j \in \mathcal{S}_i \right\},$$

where ID has a fixed length (*e.g.*, by applying padding). The algorithm's output is a random ordering of the set $\mathsf{ct}_{\mathsf{ID},i}$.

Eval$(\mathsf{ct}_{\mathsf{ID},1}, \dots, \mathsf{ct}_{\mathsf{ID},n}) \to \bigcap_{i=1}^{n} \mathcal{S}_i$**.** For each n-tuple

$$(c_{\mathsf{ID},1}, \dots, c_{\mathsf{ID},n}) \in \mathsf{ct}_{\mathsf{ID},1} \times \cdots \times \mathsf{ct}_{\mathsf{ID},n},$$

the evaluator uses $\mathsf{DE.Comb}(c_{\mathsf{ID},1}, \dots, c_{\mathsf{ID},n})$ to obtain either the message $\mathsf{ID} \| x_j$ or $\perp$. If the message starts with the expected ID, it adds x_j to the initially empty set $\mathcal{R}$.

After evaluating all tuples, the evaluator outputs the set $\mathcal{R}$.

Theorem 6. *The multi-client set intersection scheme defined above is secure under the security of the* DE *scheme.*

Proof. For $b \in \{0, 1\}$, we consider for every set element $x_{j,b} \in \bigcup_{i \in I} \mathcal{S}^*_{i,b}$ two cases:

- if $x_{j,b} \in \bigcap_{i \in I} \mathcal{S}^*_{i,b}$, $x_{j,b}$ is also contained in every client i's set $\mathcal{S}^*_{i,1-b}$;
- if $x_{j,b} \notin \bigcap_{i \in I} \mathcal{S}^*_{i,b}$, there is at least one set $\mathcal{S}^*_{k,1-b}$ which does not contain $x_{j,b}$, but an element $x_{j,1-b} \notin \bigcap_{i \in I} \mathcal{S}^*_{i,1-b}$ (and hence $x_{j,1-b} \notin \bigcap_{i \in I} \mathcal{S}^*_{i,b}$) instead.

For the elements x_j satisfying the first case, the adversary does not learn anything about b since for every client i we have that $x_j \in \mathcal{S}^*_{i,b}$ and $x_j \in \mathcal{S}^*_{i,1-b}$, while $|\mathcal{S}^*_{i,b}| = |\mathcal{S}^*_{i,1-b}|$ (remember that the set elements are randomly ordered).

For the elements $x_{j,b}$ satisfying the second case, we claim that the adversary does not learn anything about b by the security of the DE scheme. To see this, note that there exist at least two uncorrupted clients, with at least one client which did not encrypt the plaintext $\mathsf{ID}^* \| x_{j,b}$. Observe that the security of the DE scheme gives us that one cannot distinguish an encryption of a plaintext m_0 from an encryption of a plaintext m_1 as long as at most $t-1$ uncorrupted clients have encrypted the same plaintext. Combined with the fact that in our scheme we have set $t = n$ and the fact that we know that at least one uncorrupted client did not encrypt the message $\mathsf{ID}^* \| x_{j,b}$ and also that at least one uncorrupted client did not encrypt the message $\mathsf{ID}^* \| x_{j,1-b}$, we know that the encryption of the message $\mathsf{ID}^* \| x_{j,b}$ is indistinguishable from the encryption of the message $\mathsf{ID}^* \| x_{j,1-b}$. □

To improve efficiency, we can combine the above multi-client set intersection scheme with the efficient multi-client set intersection cardinality scheme. The construction for determining the cardinality can be used first to identify which ciphertext elements correspond to set elements that are in the set intersection. Next, we only have to use the evaluation algorithm of the multi-client set intersection scheme on these elements from which we know that they belong to the set intersection.

3.8 Evaluation

We have created proof-of-concept implementations[1] of the proposed 2C-FE schemes and the two MC-FE schemes for determining the cardinality of the intersection. The implementations are done in Python using the Charm library [AGM+13] at a 128 bit security level. The evaluations are done on a commodity laptop (i5-4210U@1.7 GHz, 8 GB RAM) using only a single core. In Figure 3.2 we show the time it took to run Eval on encrypted sets of varying

[1] Available at https://github.com/CRIPTIM/nipsi.

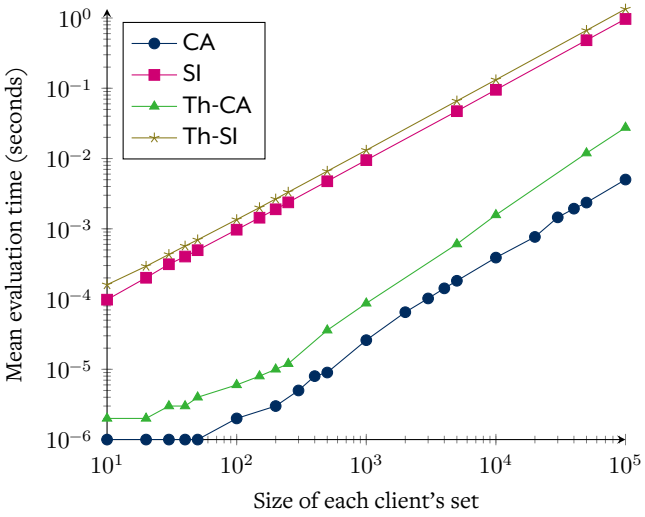
sizes. Each client encrypted a set of the same size and had 10% of their set in common with the other clients.

We see that the 2C-FE constructions can be evaluated in under a second, even for sets of 100 thousand elements in size. A lower bound of the timings is given by the 2C-FE cardinality scheme, CA, since it uses the same built-in Python algorithm that is used on plaintext data. The MC-FE constructions are polynomial in the set sizes. We evaluated the Bloom filter (BF) construction with a worst-case false positive rate of 0.001.[2] While it scales linear for fixed Bloom filter sizes, we also have to linearly increase the length of the bit strings in the set size. This results in a quadratic overall efficiency of the Eval algorithm.

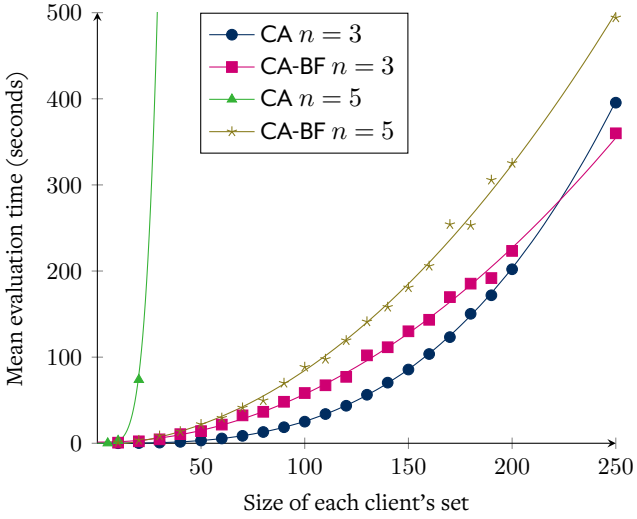
3.9 Conclusion

We initiated the study of non-interactive two-client functional encryption (2C-FE) and multi-client functional encryption (MC-FE) schemes for set intersection. We show that very efficient 2C-FE schemes can be constructed for set intersection and related set operations. Additionally, the problem of constructing non-interactive set intersection schemes for three or more clients is addressed by our MC-FE schemes from a theoretical perspective. Finally, we show the practicability of the proposed schemes using proof-of-concept implementations.

[2] An upper bound for the false positive rate of a Bloom filter resulting from the set intersection is the false positive rate of a Bloom filter with the same number of elements inserted minus the expected set intersection size. For our worst-case evaluations, we set the expected set intersection size to zero.



(a) Timings for 2C-FE schemes. In evaluating the threshold scheme, we set the threshold required to recover the set intersection to $t = 5$.



(b) Timings for MC-FE schemes with a varying number of clients n. The timings are interpolated on the domain [0, 250].

Figure 3.2. Evaluations for determining the cardinality (CA); cardinality using Bloom filters (CA-BF); set intersection (SI); and cardinality (Th-CA) and set intersection (Th-SI) in the threshold scheme.

4 Equality Tests

Vector Equality With Optional Wildcards

In the chapter above, we explore multi-client functional encryption for set operations. In this chapter, we consider another functionality to answer Research Question 1. This new functionality is for a class of predicates, known as conjunctive equality tests. To achieve this, we propose the first multi-client predicate-only encryption scheme capable of efficiently testing the equality of two encrypted vectors. Our construction can be used for the privacy-preserving monitoring of relations among multiple clients. Since both the clients' data and the predicates are encrypted, our system is suitable for situations in which this information is considered sensitive. We prove our construction plaintext and predicate private in the generic bilinear group model using random oracles, and secure under chosen-plaintext attacks with unbounded corruptions under the external decisional Diffie–Hellman assumption. Additionally, considering Research Question 2, we provide a proof-of-concept implementation that is capable of evaluating one thousand predicates defined over the inputs of ten clients in less than a minute on commodity hardware.

This chapter is based on the work "Multi-client Predicate-Only Encryption for Conjunctive Equality Tests" [KPE+17], presented at CANS.

4.1 Introduction

Predicate encryption (PE) [KSW08] is a special type of encryption that supports the evaluation of functions on encrypted data. On a conceptual level, in predicate encryption a ciphertext of a message m is associated with a descriptive value x and a decryption key sk_f with a predicate f. The decryption of a ciphertext using a key sk_f only succeeds if the predicate $f(x)$ evaluates to TRUE. Special-purpose variants of this notion include identity-based encryption (IBE) [BF01], attribute-based encryption (ABE) [SW05], and hidden

vector encryption (HVE) [BW07]. Another variant of PE is *predicate-only encryption* [KSW08; SSW09]. In predicate-only encryption, ciphertexts do not contain a message m, but merely consist of an encryption of the descriptive value x. In this case, the decryption algorithm returns the outcome of the predicate f evaluated on the predicate subject x, that is, $f(x)$.

The concept of PE can be generalized to functional encryption (FE) [BSW11; ONe10], in which the decryption of a ciphertext using a key sk_f for a (not necessarily predicate) function f does not return the original plaintext m, but the value $f(m)$ instead. More recently, Goldwasser *et al.* [GGG+14] formally defined multi-client functional encryption (MC-FE). MC-FE is a type of secret key encryption in which n distinct clients can individually encrypt a message m_i using their secret encryption key sk_i. Using a decryption key for an n-ary function f, the decryption algorithm takes as input the n ciphertexts of the clients and returns $f(m_1, \ldots, m_n)$. Although FE for generalized functionalities [GGH+13a; GGG+14] is an active field of research and of great theoretical interest, FE constructions for a restricted family of functions (such as predicates) are often far more efficient than FE schemes for arbitrary polynomially sized circuits. For example, most works in the area of MC-FE for generalized functionalities rely on inefficient primitives such as indistinguishability obfuscation ($i\mathcal{O}$) or multilinear maps.

In this work, we propose the first *multi-client* predicate-only encryption scheme. Our construction can evaluate an n-ary predicate f on the descriptive values x_i coming from n distinct clients. The type of predicates that we can evaluate using our construction is restricted to conjunctive equality tests. To put it simply, our multi-client predicate-only encryption (MC-POE) scheme is capable of testing the equality of two encrypted vectors. One of these vectors is determined by the decryption key, while the other vector is composed of ciphertexts from several distinct clients. We also provide an extension to our construction in which the decryption keys may contain wildcard components. A wildcard component in the decryption key indicates that it does not matter what the client corresponding to that vector component encrypts: any value matches the wildcard. An attentive reader familiar with the concept of HVE [BW07] will recognize the functional similarity between the two concepts. However, a crucial difference in our construction is that the ciphertext vector is composed of the ciphertexts from multiple clients, instead of being generated by a single party. A further comparison of related work is discussed in Section 4.1.2.

Our multi-client predicate-only encryption construction uses pairing-based cryptography and satisfies two distinct security notions. The first notion covers both the *attribute-hiding* [KSW08] (also referred to as *plaintext-*

privacy [SSW09]) and *predicate-privacy* [SSW09] properties of predicate encryption. Informally, these properties guarantee that an adversary can neither learn the value x of a ciphertext, nor learn the predicate from a given decryption key. Since we construct a multi-client scheme, we choose to adapt the established MC-FE security requirement [GGG+14] for our *full security* notion of multi-client predicate-only encryption. This full security notion protects against an attacker that has oracle access to both the key generation algorithm and the encryption algorithm. In the associated security game, the adversary is additionally allowed to statically corrupt clients. We prove our construction secure in the generic bilinear group model using random oracles. We also propose the (intuitively weaker) *chosen-plaintext security* notion, in which an attacker has only oracle access to the encryption algorithm, but can instead corrupt an *unbounded* number of clients. We prove our construction secure under this second notion in the standard model using the external decisional Diffie–Hellman assumption.

Our construction is designed to be simple and fast. We have implemented and analyzed our construction to evaluate whether it is efficient enough to run in practice. In our proof-of-concept implementation, clients can encrypt their values in about 2.6 ms, while decryption keys, depending on the number of vector components, can be created in less than a second. The Eval algorithm, used to evaluate the predicate on the multiple inputs, scales linearly in the number of inputs and requires only 0.10 seconds for the comparison of vectors of length 20.

4.1.1 Motivating Use Cases

Privacy-preserving monitoring over encrypted data is one of the main applications for multi-client predicate-only encryption. For example, consider the monitoring of a system comprised of various independent subsystems. We want to raise an alarm when a dangerous combination of events at the various subsystems occurs. By centrally collecting status messages of the individual systems, we can check for such situations. Such a central collection of status messages additionally avoids the need for costly interactions between the various systems. However, if these status messages are considered sensitive, the monitoring cannot be done on the cleartext messages. Multi-client predicate-only encryption overcomes this problem by allowing a monitor to evaluate an n-ary predicate over multiple ciphertexts and raise an alarm when the predicate returns TRUE.

A careful reader might realize that encryption of the status messages is not a sufficient requirement. If the monitor can check arbitrary predicates,

it can as well recover the individual plaintext status messages[1], making its encryption useless. Therefore, we have to require that another party issues the decryption keys to the monitor. Since we can consider the monitor to be a third party, it is unlikely that it is allowed to learn the predicates, making a strong case for the requirement of both plaintext privacy and predicate privacy.

The functionality of our construction is developed with the applications in the critical infrastructure (CI) domain in mind. The benefits of information sharing are widely acknowledged [PCC97], but stakeholders are still very reluctant in sharing their information with other parties [MS02; DS09; SSF16]. We give two concrete use cases.

- *Detection of coordinated attacks.* While a single failure of a system in a CI may occur occasionally, a sudden failure of multiple systems from distinct CI operators, could be an indication of a large scale cyberattack. By centrally monitoring the "failure"/"running" status messages of the CI operators, a warning can be given to the national computer emergency response team whenever a combination of systems fails, allowing further investigation of the failures. Additionally, instead of sharing just binary messages to indicate whether a system has failed, it is also helpful to share and monitor *cyberalert levels*. These cyberalert levels from different clients are used to get an improved situational overview [LK15b].

- *Monitoring of dependencies among CI operators.* There exist many dependencies among various CIs [LNK+09], making it possible for disruptions to easily propagate from one infrastructure to another [CLO+06]. By timely reporting status messages on supply, a central authority can determine whether supply will meet demand and otherwise instruct parties to prepare their backup resources. Similarly, the sharing of compliance status (*e.g.*, whether they can be met or not) can be used to take the right security measures at another party [LK15b].

4.1.2 Related Work

A multi-*input* functional encryption (MI-FE) [GGG+14] scheme is an FE scheme that supports the computation of functions over multiple encrypted inputs. Examples of special-purpose MI-FE include property-preserving encryption [PR12], such as for ordering [BLR+15; CLW+16] or equality [YTH+10],

[1]For example, the monitor could create a decryption key for a predicate evaluation of a single message, *e.g.*, $f(x_1, \ldots, x_n) = \text{TRUE}$ if and only if $x_1 = 0$.

and multi-input inner product encryption (MI-IPE) [AGR+17]. The MI-IPE scheme by Abdalla *et al.* [AGR+17] is capable of computing the inner product of two vectors, *i.e.*, the decryption algorithm returns a scalar. This should not be confused with an inner-product *predicate* encryption scheme where *predicates* (with a TRUE/FALSE result) can be evaluated by an inner product. A private-key, multi-*client* FE (MC-FE) scheme [GKL+13; GGG+14] is a variant of MI-FE. There are two key differences between the two notions. Firstly, MC-FE requires that the ciphertexts for the function inputs are generated by individual *distinct* parties, while in MI-FE it is allowed to have only a single encryptor for all the inputs. Secondly, in MC-FE the ciphertexts are associated with a *time-step* [GGG+14] or *identifier*. Such an identifier is used to prevent mix-and-match attacks: decryption only works when all ciphertexts are associated with the same identifier.

Although not recognized as such, several special-purpose MC-FE schemes have already been proposed in literature. Shi *et al.* [SCR+11] propose a construction for the privacy-preserving aggregation of time-series data. Their construction allows a central party to compute and learn the sum over encrypted numbers, without learning the individual numbers themselves. Decentralized multi-authority attribute-based encryption (MA-ABE) [LW11] can also be considered a form of MC-FE. In MA-ABE, several decryption keys, issued by different authorities and associated with an identifier, need to be combined to decrypt a single ciphertext. The similarity becomes apparent once we swap the roles of the ciphertext and decryption keys.

Wildcards have been used in PE before by Abdalla *et al.* [ACD+06] in IBE and by Boneh and Waters [BW07] in HVE. These works differ from our work in several aspects. Most importantly, our construction is a multi-client variant instead of single-client. If we would apply a single-client construction in a multi-client setting, we would leak the individual predicate results for each party. Secondly, we achieve both plaintext privacy and predicate privacy, which is known to be impossible to accomplish in the public-key setting [SSW09] ([ACD+06; BW07] are in the public-key setting). Finally, we look at predicate-only encryption, not at regular PE in which the ciphertexts may also contain an encrypted *payload* message.

Numerous PE schemes are used for searchable encryption (SE) [BHJ+14]. However, we see no great benefit in applying MC-POE as an SE scheme. An MC-POE scheme enables us to compute a predicate over multiple inputs from several explicitly chosen clients. In SE, this would correspond to a search over documents where the query specifies which keywords have to be set by which parties. This is also the reason why existing multi-*writer* [BHJ+14] schemes, do not consider searching over documents using queries which, for

example, specify that party p_1 should have added keyword w_1, while party p_2 should have added keyword w_2.

4.2 Multi-client Predicate-Only Encryption

A multi-client predicate-only encryption (MC-POE) scheme is a collection of the following four polynomial-time algorithms.

Setup$(1^\lambda, n)$**.** This algorithm defines the public parameters pp, a master secret key msk, and the encryption keys usk_i for every client $1 \leq i \leq n$. The algorithm also defines the finite message space $\mathcal{M}^n$ and the predicate family $\mathcal{F}$, which predicates are efficiently computable on $\mathcal{M}^n$.

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, x_i)$**.** A client i can encrypt a value $x_i \in \mathcal{M}$ using its encryption key usk_i and an identifier ID. Different clients can use the same identifier. However, each client can only use an identifier at most once. The algorithm returns a ciphertext $\mathsf{ct}_{\mathsf{ID},i}$. We usually omit the index ID when there is no ambiguity. Furthermore, we introduce the following simplification of notation for a set of ciphertexts associated with the same ID: For an ordered set $\mathcal{S} \subseteq \{1, \dots, n\}$ of indices, we write the set of ciphertexts $\{\, \mathsf{Encrypt}(\mathsf{usk}_j, \mathsf{ID}, x_j) \mid j \in \mathcal{S} \,\}$ as $\mathsf{Encrypt}(\mathsf{usk}_\mathcal{S}, \mathsf{ID}, \mathbf{x}_\mathcal{S})$. If $\mathcal{S} = \{1, \dots, n\}$, we simply write $\mathsf{Encrypt}(\mathsf{usk}, \mathsf{ID}, \mathbf{x})$ or $\mathsf{ct}_\mathbf{x}$.

KeyGen(msk, f)**.** The key generator can create an evaluation key, also termed *token*, for predicate $f \in \mathcal{F}$ using msk. The algorithm returns the token esk_f.

Eval$(\mathsf{esk}_f, \mathsf{ct}_\mathbf{x})$**.** The Eval algorithm requires a vector of ciphertexts $\mathsf{ct}_\mathbf{x}$ and a token esk_f as input. The algorithm outputs a Boolean value.

Correctness A multi-client predicate-only encryption scheme is *correct* if $\mathsf{Eval}(\mathsf{esk}_f, \mathsf{ct}_\mathbf{x}) = f(\mathbf{x})$. Formally, we require for all $n \in \mathbb{N}$, $\mathbf{x} \in \mathcal{M}^n$, and $f \in \mathcal{F}$,

$$\Pr\left[\mathsf{Eval}(\mathsf{ct}_\mathbf{x}, \mathsf{esk}_f) \neq f(\mathbf{x}) \colon \begin{array}{r} (\mathsf{pp}, \mathsf{msk}, \{\mathsf{usk}_i\}) \leftarrow \mathsf{Setup}(1^\lambda, n) \\ \mathsf{ct}_\mathbf{x} \leftarrow \mathsf{Encrypt}(\mathsf{usk}, \mathsf{ID}, \mathbf{x}) \\ \mathsf{esk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \end{array}\right]$$

is negligible in the security parameter λ, where the probability is taken over the coins of Setup, $\mathsf{Encrypt}$, and KeyGen.

Note that we do not impose any restriction on the output of Eval if it operates on messages encrypted under different identifiers.

4.2.1 Multi-client Predicate-Only Encryption Security

A commonly considered security game for private-key functional encryption is an indistinguishability-based notion under which the adversary may query both the Encrypt and the KeyGen oracles [KSW08; SSW09; GGG+14] (*cf.* Definition 6). Since our MC-POE is a special case of MC-FE, we start from the security notion from Goldwasser *et al.* [GGG+14]. However, they only consider the indistinguishability of plaintexts (*plaintext privacy* [KSW08; SSW09]) and not of functions (*function* or *predicate privacy* [SSW09; BS15]) in their security definition. In the following *full security* notion, we combine the plaintext-privacy and predicate-privacy notions, similarly to Shen, Shi, and Waters [SSW09].

Because an evaluation of a predicate on a set of messages reveals some information about the messages in relation to the predicate (and *vice versa*), we cannot allow the adversary to query for all combinations of messages and predicates. For example, an adversary can distinguish an encryption of message $\mathbf{x}_0$ from an encryption of $\mathbf{x}_1$ if it has a token for a predicate f such that $f(\mathbf{x}_0) \neq f(\mathbf{x}_1)$. Even if we require $f(\mathbf{x}_0) = f(\mathbf{x}_1)$ for all predicates f that the adversary queried, a similar situation can still appear. To see this, consider an adversary corrupting client i so that it can encrypt any message m_i as ith input. This means that the adversary can also trivially distinguish the two messages if there exists a value m_i, such that if it replaces the ith input of $\mathbf{x}_0$ and $\mathbf{x}_1$ by m_i (resulting in inputs $\mathbf{x}'_0$ and $\mathbf{x}'_1$ respectively), the predicate has different outputs, *i.e.*, $f(\mathbf{x}'_0) \neq f(\mathbf{x}'_1)$. Likewise, we also have to require that the predicates f_0 and f_1 yield the same result on a queried input $\mathbf{x}$, even if the adversary replaces some of the corrupted clients' inputs by another value.

In our security definition, we use the term *static corruptions* to indicate that the adversary announces the corrupted clients at the beginning of the game and cannot corrupt additional clients during the rest of the game. We let I be the set of indices of the uncorrupted clients, and similarly, indicate the indices of the corrupted clients by the set $\bar{I}$. Recall that we use the notation $\mathbf{x}_I$ to denote the subvector of $\mathbf{x}$ containing only the components from the set I. We denote with $f(\mathbf{x}_I, \cdot)$ a predicate f with the pre-filled inputs $\mathbf{x}_I$.

Definition 8 (Adaptive Full Security)**.** A multi-client predicate-only encryption scheme is *adaptive full secure under static corruptions* if every probabilistic polynomial time adversary $\mathcal{A}$ has at most a negligible advantage in winning the following game.

Initialization The adversary $\mathcal{A}$ submits a set of indices $\bar{I}$ to the challenger. We define the complement set $I = \{1, \dots, n\} \setminus \bar{I}$.

Setup The challenger runs $\mathsf{Setup}(1^\lambda, n)$ to get the pp, msk, and $\{\mathsf{usk}_i\}_{1 \leq i \leq n}$.

It gives the public parameters pp and corrupted clients' keys $\{ \mathsf{usk}_i \mid i \in \bar{I} \}$ to the adversary.

Query 1 The adversary $\mathcal{A}$ may query the challenger for ciphertexts or tokens.

- **Ciphertext** In case of a ciphertext query for (i, ID, x_i), the challenger returns $\mathsf{ct}_{\mathsf{ID},i} \leftarrow \mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{ID}, x_i)$.
- **Token** For a token query for f, the challenger returns the evaluation key $\mathsf{esk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$.

Challenge The challenger picks a random bit b. The adversary can either request a ciphertext challenge or a token challenge.

- **Ciphertext** In case of a ciphertext challenge, the adversary sends $(\mathsf{ID}^*, \mathbf{x}^*_{0,I}, \mathbf{x}^*_{1,I})$ to the challenger. The challenger answer the query with the challenge ciphertext $\mathsf{Ch}_I \leftarrow \mathsf{Encrypt}(\mathsf{usk}_I, \mathsf{ID}^*, \mathbf{x}^*_{b,I})$.
- **Token** In case of a token challenge, the adversary sends (f^*_0, f^*_1) to the challenger. The challenger returns the challenge token $\mathsf{Ch} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f^*_b)$.

Query 2 The adversary may query the challenger again, similar to Query 1.

Guess The adversary outputs its guess $b' \in \{0, 1\}$ for the bit b.

We say that adversary $\mathcal{A}$ wins the game, if $b' = b$ and

- in case of a ciphertext challenge, $\mathcal{A}$ did not query for a ciphertext using identifier ID^* in any of the two query phases, nor query for a predicate f, such that $f(\mathbf{x}^*_{0,I}, \cdot) \neq f(\mathbf{x}^*_{1,I}, \cdot)$;
- in case of a token challenge, $\mathcal{A}$ did not query for (i, ID, x_i), for uncorrupted clients $i \in I$, such that it can combine these inputs x_i for the same ID, into a vector $\mathbf{x}_I$, where $f^*_0(\mathbf{x}_I, \cdot) \neq f^*_1(\mathbf{x}_I, \cdot)$.

Note that in the above defined game, in case of a ciphertext challenge, the challenger only returns challenge ciphertexts for the uncorrupted clients. The adversary can still evaluate predicates on the received challenge by generating the ciphertext values for the corrupted clients using their encryption keys.

It is important to realize that the challenger can decide whether the adversary wins the game or not in polynomial time. This is possible because the adversary $\mathcal{A}$ can only query for a polynomial number of ciphertexts and tokens. Moreover, the challenger is able to efficiently check if $f(\mathbf{x}_I, \cdot) = f'(\mathbf{x}'_I, \cdot)$ as both n and $\mathcal{M}^n$ are finite and fixed by $\mathsf{Setup}(1^\lambda, n)$.

Definition 9 (Selective Full Security)**.** The definition of a *selective full secure under static corruptions* multi-client predicate-only encryption scheme is similar to the adaptive full security notion of Definition 8. The difference between the two, is that in selective security game, the challenge request (*i.e.*, either $(\mathsf{ID}^*, \mathbf{x}^*_{0,I}, \mathbf{x}^*_{1,I})$ or (f^*_0, f^*_1)) is announced during Initialization.

As explained before, the full security definition actually defines two security notions. We say that an MC-POE scheme is *adaptive* (*selective*) *plaintext private* if no adversary can win the adaptive (selective, respectively) full security game with a ciphertext challenge. Similarly, an MC-POE scheme is *adaptive* (*selective*) *predicate private* if no adversary can win the adaptive (selective, respectively) full security game with a token challenge.

Chosen-Plaintext Security The definition of full security is very strong as it allows an adversary to query for both ciphertexts and tokens. This is similar to the chosen-ciphertext attack (CCA) security notion used in public-key cryptography, where the adversary can query both the encryption and decryption[2] oracle. To accommodate for a different attacker model, we define a *chosen-plaintext* security notion, where the adversary only has access to the encryption oracle and is asked to distinguish between two ciphertexts. Such a notion is similar to chosen-plaintext attack (CPA) security as defined in public-key cryptography and is also related to the *offline security* notion of Lewi and Wu [LW16], in which an attacker has only access to ciphertexts and not to decryption keys. To make our notion stronger, we give the adversary access to all clients' encryption keys (but not to the internal randomness of the clients).

Definition 10 (Chosen-Plaintext Security)**.** A multi-client predicate-only encryption scheme is *chosen-plaintext secure under unbounded corruptions* if any probabilistic polynomial time algorithm $\mathcal{A}$ has at most a negligible advantage in winning the following game.

Setup The challenger runs $\mathsf{Setup}(1^\lambda, n)$ to get the pp, msk, and $\{\mathsf{usk}_i\}_{1\le i\le n}$. It gives the public parameters pp and all clients' keys $\{\mathsf{usk}_i\}_{1\le i\le n}$ to the adversary. Note that the adversary $\mathcal{A}$ can encrypt any message x_i for identifier ID using the key usk_i by computing $\mathsf{Encrypt}(\mathsf{usk}_i, \mathsf{ID}, x_i)$.

Challenge The adversary sends the challenge request $(\mathsf{ID}^*, \mathbf{x}^*_0, \mathbf{x}^*_1)$ to the challenger. The challenger picks a random bit b and returns the challenge ciphertext $\mathsf{Encrypt}(\mathsf{usk}, \mathsf{ID}^*, \mathbf{x}^*_b)$ to the adversary.

[2] In MC-POE, an adversary can use a token and the public Eval algorithm to learn more about the encrypted plaintext.

Guess The adversary outputs its guess $b' \in \{0,1\}$ for the bit b.

We say that adversary $\mathcal{A}$ wins the game if $b' = b$.

Observe that in this game the adversary is given *every* client's private key. This security requirement is quite strong and corresponds to a following situation: Even if an attacker compromises a client and steals its encryption keys, it *remains* hard for the attacker to determine the plaintexts of the ciphertexts created *before and after* the compromise.

4.3 Our Construction

We construct a multi-client predicate-only encryption scheme for the functionality of a conjunctive equality test. To evaluate if n messages $x_1, \ldots, x_n$, encrypted by distinct clients, equal the values $y_1, \ldots, y_n$, we evaluate the predicate

$$\mathsf{Match}(\mathbf{x}, \mathbf{y}) = \begin{cases} \text{TRUE} & \text{if } \bigwedge_{i=1}^{n} (x_i = y_i), \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

As discussed in Section 4.1.1, this functionality turns out to be surprisingly useful in the domain of critical infrastructure protection. In this setting, a monitor combines the ciphertexts associated with the same identifier and evaluates all its tokens (corresponding to various predicates) on the ciphertext vector to see if there is a match. If a match is found, the monitor may raise an alarm or take other appropriate actions. A schematic overview of relations among all parties of such a multi-client monitoring system is shown in Figure 4.1.

We now describe our multi-client predicate-only encryption construction for conjunctive equality tests over multiple clients.

Setup$(1^\lambda, n)$**.** Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}_3(1^\lambda)$ be the parameters for a Type 3 asymmetric bilinear group. Choose a pseudorandom permutation (PRP) $\pi\colon \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ for message space $\mathcal{M} \subseteq \mathbb{Z}_p$ and a cryptographic hash function $H\colon \{0,1\}^* \to \mathbb{G}_1$. The bilinear group parameters together with both functions form the public parameters. To generate the keys, select $\alpha_i, \gamma_i \xleftarrow{R} \mathbb{Z}_p^*$ and $\beta_i \xleftarrow{R} \mathcal{K}$ for $1 \leq i \leq n$. The master secret key is

$$\mathsf{msk} = \left\{ (g_2^{\alpha_i}, \beta_i, g_2^{\gamma_i}) \right\}_{i=1}^{n}.$$

The secret encryption key for client i is

$$\mathsf{usk}_i = (g_1^{\alpha_i}, \beta_i, \gamma_i).$$

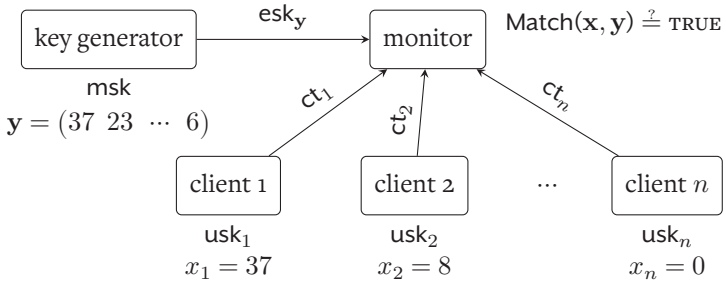


Figure 4.1. In this example of a multi-client monitoring system, there are n distinct clients (with keys $\mathsf{usk}_1, \ldots, \mathsf{usk}_n$) that determine the values $x_1, \ldots, x_n$. The monitor computes the functionality $\mathsf{Match}(\mathbf{x}, \mathbf{y})$ using the encrypted values $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ and an evaluation token $\mathsf{esk}_{\mathbf{y}}$. The monitor is only able to compute the functionality if all clients encrypted their value x_i using the same identifier ID (not shown in the figure).

Encrypt$(\mathsf{usk}_i, \mathsf{ID}, x_i)$**.** Client i can encrypt its message $x_i \in \mathcal{M}$ for identifier ID using usk_i and $r_i \xleftarrow{R} \mathbb{Z}_p^*$,

$$\mathsf{ct}_i = \big(H(\mathsf{ID}), g_1^{r_i}, g_1^{\alpha_i \pi(\beta_i, x_i) r_i} H(\mathsf{ID})^{\gamma_i}\big).$$

KeyGen$(\mathsf{msk}, \mathbf{y})$**.** The token generator can encrypt a vector $\mathbf{y} \in \mathcal{M}^n$ using its key msk. Choose $u_i \xleftarrow{R} \mathbb{Z}_p^*$ for $1 \le i \le n$ and output

$$\mathsf{esk}_{\mathbf{y}} = \left(\{\, g_2^{u_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i} \mid 1 \le i \le n \,\}, \prod_{1 \le i \le n} (g_2^{\gamma_i})^{u_i} \right).$$

Eval$(\mathsf{esk}_{\mathbf{y}}, \{\mathsf{ct}_i\}_{1 \le i \le n})$**.** Output the result of the evaluation test

$$\prod_{1 \le i \le n} e\big(g_1^{\alpha_i \pi(\beta_i, x_i) r_i} H(\mathsf{ID})^{\gamma_i}, g_2^{u_i}\big) \stackrel{?}{=} \prod_{1 \le i \le n} e\big(g_1^{r_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i}\big) \cdot e\big(H(\mathsf{ID}), \prod_{1 \le i \le n} (g_2^{\gamma_i})^{u_i}\big).$$

4.3.1 Correctness

Correctness follows from the definition of Eval. We remark that the output of Eval is completely determined by $\sum_{1 \le i \le n} (\pi(\beta_i, x_i) - \pi(\beta_i, y_i)) \stackrel{?}{=} 0$. Since the function π is a PRP, the probability of $\mathsf{Eval}(\mathsf{esk}_{\mathbf{y}}, \mathsf{ct}_{\mathbf{x}}) \neq \mathsf{Match}(\mathbf{x}, \mathbf{y})$ is negligible.

4.3.2 Security

To get an intuition for the security of our construction, observe that the clients' messages itself are first encrypted using the PRP π. By using the output of the PRP as an exponent and randomizing it with the value r, we create a probabilistic encryption of the message. The PRP's randomized output also prevents malleability attacks. Similarly, the vector components of the vector $\mathbf{y}$ are individually encrypted in a similar way. Because part of the clients' keys (*i.e.*, $g_1^{\alpha_i}$) and the master secret key (*i.e.*, $g_2^{\alpha_i}$) reside in different groups, it is hard for a client to create a token and hard for the token generator to create a ciphertext.

The formal security analysis can be found in Section 4.4. We prove our construction selective plaintext private and adaptive predicate private. Additionally, we prove the chosen-plaintext security property of the construction. Plaintext and predicate privacy are proven in the generic group model using random oracles. This combination of models has been successfully applied in other works before [Sma01; CMZ14]. Chosen-plaintext security can be proven in the standard model and under the decisional Diffie–Hellman (DDH) assumption in group $\mathbb{G}_1$. We formulate the following two theorems.

Theorem 7. *Let $\mathcal{A}$ be an arbitrary probabilistic polynomial time adversary having oracle access to the group operations and the encryption and token generation algorithms, while it is bounded in receiving at most q distinct group elements. The adversary $\mathcal{A}$ has at most an advantage of $O(q^2 \,/\, p)$ in winning either the selective plaintext-privacy (see Definition 9) or the adaptive predicate-privacy game (see Definition 8) in the random oracle model.*

Theorem 8. *The construction presented above is chosen-plaintext secure with an unbounded number of corruptions (Definition 10) under the DDH assumption in group $\mathbb{G}_1$.*

Both plaintext privacy and predicate privacy are proven secure through a series of hybrid games. In every game hop, a component of the challenge vector (either the ciphertext or token challenge vector) is replaced by a random one. In the final game, once all components are replaced by random elements, no adversary can gain an advantage since it is impossible to distinguish a random vector from another random one.

However, in the selective plaintext-privacy game, not *every* component of the challenge vector can be replaced by a random component. If a component $x^*_{b,i}$ of the challenge vector $\mathbf{x}^*_b$ is deterministic, *i.e.*, the challenge inputs were the same for that component, $x^*_{0,i} = x^*_{1,i} = m$, the adversary may query for a token to match this single component for the value $y_i = m$.

Note that if this component is replaced by a random element, Match will, with overwhelming probability, return FALSE, while it should have returned TRUE. Hence, the deterministic components of the challenge vector have to remain untouched in every game hop. This implies that the number of game hops depends on the challenge inputs, requiring the challenger to know the challenge inputs *a priori*. This limitation does not appear for predicate privacy, making it possible to prove adaptive security instead.

4.3.3 Extension Allowing Wildcards

Although a construction for the described conjunctive equality matching functionality would suffice, it may be very inefficient when a predicate is defined over a subset of the clients' inputs. For example, suppose the token generator has a predicate for which it actually does not care what client i sends. Now, if we have only conjunctive equality matching, we would need to create a token for every possible message that client i can send. Besides that this will be very inefficient if client i could send many different messages, it would also reveal whenever client i has sent the same values multiple times: Whenever a client sends the same value multiple times, the same token will match multiple times as well!

We can extend our construction with the ability to test for the equality of vectors with the additional feature that the predicate vector $\mathbf{y}$ can now contain wildcard components. Such a wildcard component matches against any value of the corresponding ciphertext component. This makes the testing functionality similar to the one used in HVE [BW07], however, our system combines the ciphertexts from multiple clients. Formally, the clients encrypt their messages from the message space $\mathcal{M} \subseteq \mathbb{Z}_p$, where the token generator uses the space $\mathcal{M}^* = \mathcal{M} \cup \{\star\}$. The multi-client predicate-only encryption construction now evaluates the function

$$\mathsf{Match}^\star(\mathbf{x}, \mathbf{y}) = \begin{cases} \text{TRUE} & \text{if } \forall i\text{: } (x_i = y_i) \vee (y_i = \star), \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

To achieve this additional functionality, we have to change the KeyGen and Eval algorithms, the other algorithms remain unchanged.

KeyGen*(msk, $\mathbf{y}$). The token generator can encrypt a predicate vector $\mathbf{y} \in (\mathcal{M}^*)^n$ using the master secret key msk. Let $\mathcal{S}_\mathbf{y}$ be the set of indices of the non-wildcard components of the vector $\mathbf{y}$. Choose $u_i \xleftarrow{R} \mathbb{Z}_p^*$ for $i \in \mathcal{S}_\mathbf{y}$ and output

$$\mathsf{esk}_\mathbf{y} = \left(\left\{ g_2^{u_i}, g_2^{\alpha_i \pi(\beta_i, y_i) u_i} \mid i \in \mathcal{S}_\mathbf{y} \right\}, \prod_{i \in \mathcal{S}_\mathbf{y}} (g_2^{\gamma_i})^{u_i} \right).$$

Eval*($\mathsf{esk}_\mathbf{y}, \{\mathsf{ct}_i\}_{i\in\mathcal{S}_\mathbf{y}}$). Output the result of the test

$$\prod_{i\in\mathcal{S}_\mathbf{y}} e\big(g_1^{\alpha_i\pi(\beta_i,x_i)r_i} H(\mathsf{ID})^{\gamma_i}, g_2^{u_i}\big) \stackrel{?}{=} \prod_{i\in\mathcal{S}_\mathbf{y}} e\big(g_1^{r_i}, g_2^{\alpha_i\pi(\beta_i,y_i)u_i}\big) \cdot e\big(H(\mathsf{ID}), \prod_{i\in\mathcal{S}_\mathbf{y}} (g_2^{\gamma_i})^{u_i}\big).$$

In this adapted construction, the wildcards are made possible by allowing the token generator to specify which clients need to contribute a ciphertext before one can evaluate the predicate over the subset of clients. This idea is encoded in the token by the value $\prod_{i\in\mathcal{S}_\mathbf{y}}(g_2^{\gamma_i})^{u_i}$ and in the ciphertext by the value $H(\mathsf{ID})^{\gamma_i}$. The latter also prevents the monitor to combine ciphertext for different identifiers.

The addition of wildcards to the scheme should be mainly considered an efficiency improvement, rather than a security improvement, although the ciphertext security actually slightly improves when one uses wildcards, as the wildcard components do not leak any information about the matched ciphertext (discussed above). However, we point out that this adapted construction is not predicate private. In fact, if wildcards are used in the proposed construction, the token would leak their positions: by looking at a token, it is possible to tell which components encode a wildcard. But, if we accept this fact, yet still want to assure that no other information is leaked, we can define a *restricted* predicate-privacy game. In this restricted game, we restrict the adversary to only provide challenge inputs with wildcards in the same position, *i.e.*, we require for challenge inputs $f_0^* = \mathbf{y}_0^*$, $f_1^* = \mathbf{y}_1^*$ that for all $1 \leq i \leq n$, $y_{0,i} = \star \iff y_{1,i} = \star$.

It is trivial to see that changing the KeyGen or Eval algorithm does not influence the chosen-ciphertext security. In Section 4.4 we give the security proofs for the construction with wildcards.

4.3.4 Efficiency

Since the Encrypt and KeyGen algorithms do not use any expensive pairing operations, they can efficiently run on less powerful hardware. For the Encrypt algorithm it is only needed to compute the PRP π and three modular exponentiations. The computational complexity of KeyGen* depends on the number of non-wildcard components in the predicate. For every non-wildcard component one evaluation of the PRP π and three modular exponentiations are needed.

The Eval algorithm is the only algorithm that requires pairings. To evaluate a token with n non-wildcard components, $2n + 1$ pairings are required.

In the next section we discuss a concrete implementation of the construction and evaluate its performance.

4.4 Security Proofs

4.4.1 Selective Plaintext and Adaptive Predicate Security

We prove Theorem 7, stating that the construction without wildcards is secure, by using the following lemma and by proving that the construction with wildcards is selective plaintext private and restricted adaptive predicate private. Recall that the restricted predicate-private game is almost identical to our predicate-private game. However, in the restricted game, we additionally require $y^*_{0,i} = \star \iff y^*_{1,i} = \star$ for the challenge inputs $\mathbf{y}^*_0, \mathbf{y}^*_1$.

Lemma 1. *If the construction* with *wildcards is selective plaintext private and* restricted *adaptive predicate private, then the construction* without *wildcards is selective plaintext private and adaptive predicate private.*

Proof. First, let us look at the selective plaintext privacy. Assume $\mathcal{A}$ is a probabilistic polynomial time adversary, having a non-negligible advantage in winning the selective plaintext-privacy game without wildcards. It is clear that $\mathcal{A}$ is also an adversary that has an identical, non-negligible, advantage in winning the selective plaintext-privacy game with wildcards (however, it chooses not to use any). This contradicts with the given statement that no such adversary exists.

For the other part, assume that $\mathcal{A}$ is a probabilistic polynomial time adversary, making no wildcard queries, and having a non-negligible advantage in winning the predicate-privacy game. Note that $\mathcal{A}$ is also an adversary that has an identical, non-negligible, advantage in winning the predicate-privacy game with wildcards (however, it chooses not to use any). Specifically, since $\mathcal{A}$ chooses its challenge inputs without wildcards, $\mathcal{A}$ also satisfied the extra requirement in the restricted predicate-privacy game. □

We now give a proof for both selective plaintext privacy as well as restricted predicate privacy for the construction with wildcards.

Proof (sketch). We first define the generic group model setting and all oracle interactions, including the oracles for encryption and token generation.

Generic group model. Let ϕ_1, ϕ_2, ϕ_T be distinct random injective mappings from the domain $\mathbb{Z}_p$ to $\{0,1\}^\lambda$, where $\lambda > 3 \log p$. We write $\mathbb{G}_1$ for $\{\, \phi_1(x) \mid x \in \mathbb{Z}_p \,\}$, $\mathbb{G}_2$ for $\{\, \phi_2(x) \mid x \in \mathbb{Z}_p \,\}$, and $\mathbb{G}_T$ for $\{\, \phi_T(x) \mid x \in \mathbb{Z}_p \,\}$. The adversary is given access to an oracle to compute the group actions on $\mathbb{G}_1$,

$\mathbb{G}_2$, and $\mathbb{G}_T$. Additionally, it is given access to an oracle capable of computing a non-degenerate bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Lastly, we also define a random oracle to model the hash function $H\colon \{0,1\} \to \mathbb{G}_1$.

Instead of writing $\phi_1(x)$, we write g_1^x. Similarly, we write g_2^x for $\phi_2(x)$ and $e(g_1, g_2)^x$ for $\phi_T(x)$.

Hash oracle H. The challenger keeps track of oracle queries it received before by maintaining a table. If it has not received an oracle query for the value ID before, it chooses a random value $t_{\mathsf{ID}} \in \mathbb{Z}_p$ and stores this value in its table. It returns the value $g_1^{t_{\mathsf{ID}}}$ to the querier.

Game interactions. The adversary's first interaction with the challenger is to receive the group parameters and the secret keys of the corrupted clients.

Setup The challenger chooses $\alpha_i, \gamma_i \xleftarrow{R} \mathbb{Z}_p^*$ and $\beta_i \xleftarrow{R} \mathcal{K}$ for $1 \leq i \leq n$, just like in the actual scheme. It also defines the secret keys usk_i and master secret key msk according to the scheme.

Corruptions The adversary submits its choices for the corrupted clients $\bar{I}$ to the challenger. In the selective plaintext-privacy game, the adversary additionally submits its challenge inputs $(\mathsf{ID}^*, \mathbf{x}^*_{0,I}, \mathbf{x}^*_{1,I})$. The challenger gives the secret keys $\mathsf{usk}_{\bar{I}}$ of the corrupted clients to the adversary.

Queries The adversary interacts with the challenger by asking the challenger to encrypt a messages or to generate a token for some predicate. To be able to refer to a specific query later on in the proof, we label every query with a query number. Let j represent this query number.

Encrypt The challenger answers valid $\mathsf{Encrypt}$ queries for a message $x_i^{(j)}$ for client i and identifier $\mathsf{ID}^{(j)}$ similar as in the scheme. It chooses $r_i^{(j)} \xleftarrow{R} \mathbb{Z}_p^*$ and returns the ciphertext $\mathsf{ct}_{i,\mathsf{ID}}^{(j)}$,

$$\left(g_1^{t_{\mathsf{ID}^{(j)}}}, g_1^{r_i^{(j)}}, g_1^{\alpha_i \pi(\beta_i, x_i^{(j)}) r_i^{(j)}} g_1^{t_{\mathsf{ID}^{(j)}} \gamma_i} \right).$$

KeyGen* Token queries for $\mathbf{y}^{(j)}$ are answered according to the scheme as well. The challenger chooses $u_i^{(j)} \xleftarrow{R} \mathbb{Z}_p^*$ for $i \in \mathcal{S}_{\mathbf{y}^{(j)}}$ and returns the token $\mathsf{esk}_{\mathbf{y}}^{(j)}$,

$$\left(\left\{ g_2^{u_i^{(j)}}, g_2^{\alpha_i \pi(\beta_i, y_i^{(j)}) u_i^{(j)}} \mid i \in \mathcal{S}_{\mathbf{y}} \right\}, \prod_{i \in \mathcal{S}_{\mathbf{y}}} g_2^{u_i^{(j)} \gamma_i} \right),$$

to the adversary.

Proof structure. We prove both selective plaintext privacy and restricted adaptive predicate privacy through a series of hybrid games.

For selective plaintext privacy the number of games depends on the number of differentiating components of the challenge inputs—hence the *selective* game type. Let $\overline{X}$ denote the set of indices where the components of $\mathbf{x}_0^*$ differ from $\mathbf{x}_1^*$, $\overline{X} = \{\, i \mid x_{0,i}^* \neq x_{1,i}^* \,\}$. Let game k be identical to the original game, except that in the challenge phase now the first $k-1$ components of $\overline{X}$ in the returned challenge vector are chosen at random. Note that game $k = 1$ is identical to the original game and that in game $k = |\overline{X}|$ not even an unbounded adversary is able to gain an advantage in winning the game.

For restricted adaptive predicate privacy, we assume w.l.o.g. that $\mathbf{y}_{0,I}^* \neq \mathbf{y}_{1,I}^*$, because if $\mathbf{y}_{0,I}^* = \mathbf{y}_{1,I}^*$, the adversary would not be able to gain an advantage in the game since this implies $\mathbf{y}_0^* = \mathbf{y}_1^*$. Note that this means that the result of $\mathsf{Match}^\star$ with any allowed ciphertext vector will be FALSE. We define game k identical to the original game, except that in the challenge phase now the first $k-1$ components of the returned challenge vector are chosen at random. Note that game $k = 1$ is identical to the original game and that in game $k = n$ not even an unbounded adversary is able to gain an advantage in winning the game.

For both the selective plaintext-privacy as well as the restricted adaptive predicate-privacy game, we show that an adversary has at most an advantage of $O(q^2 \,/\, p)$ in distinguishing between game k and game $k+1$. Furthermore, we use another hybrid game to change to a real-or-random based challenge instead of a left-or-right based challenge. It is not difficult to see that an adversary gaining an advantage ϵ in the left-or-right based game, gains an advantage of at least $\epsilon \,/\, 2$ in the real-or-random based game.

Challenges. Since we changed the game to a real-or-random based game, the challenge phase changes slightly. The challenger now chooses a bit $b \xleftarrow{R} \{0,1\}$ that is used to determine whether to return the encryption of the submitted value or a random one. In case of the selective plaintext-privacy game, the adversary submits a vector $\mathbf{x}_I^{(c)}$ together with an identifier $\mathsf{ID}^{(c)}$ to the challenger. In case of the restricted predicate-privacy game, the adversary submits a vector $\mathbf{y}^{(c)}$ to the challenger. The challenger chooses values $\nu_i, \nu_i' \xleftarrow{R} \mathbb{Z}_p^*$ for $1 \leq i \leq n$. For a ciphertext challenge it returns the challenge

$$\mathsf{ct}_{\mathsf{Ch}} = \left\{ \left(g_1^{t_{\mathsf{ID}^{(c)}}}, g_1^{\nu_i}, \mathsf{ct}'_{\mathsf{Ch},i} \right) \mid i \in I \right\},$$

where

$$\mathsf{ct}'_{\mathsf{Ch},k} = \begin{cases} g_1^{\nu_k \alpha_k \pi(\beta_k, \nu'_k) + t_{\mathsf{ID}^{(c)}} \gamma_k} & \text{if } b = 0 \\ g_1^{\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k} & \text{if } b = 1. \end{cases}$$

For a token challenge, it returns the challenge

$$\mathsf{esk}_{\mathsf{Ch}} = \left(\left\{ \left(g_2^{\nu_i}, \mathsf{esk}'_{\mathsf{Ch},i} \right) \mid i \in \mathcal{S}_{\mathbf{y}} \right\}, \prod_{i \in \mathcal{S}_{\mathbf{y}}} g_2^{\nu_i \gamma_i} \right),$$

where, if $k \in \mathcal{S}_{\mathbf{y}}$,

$$\mathsf{esk}'_{\mathsf{Ch},k} = \begin{cases} g_2^{\nu_k \alpha_k \pi(\beta_k, \nu'_k)} & \text{if } b = 0 \\ g_2^{\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})} & \text{if } b = 1. \end{cases}$$

Indistinguishability. We now show that an adversary has at most a negligible advantage of $O(q^2 / p)$ in distinguishing between game k and game $k+1$, *i.e.*, it is unable to distinguish $g_1^{\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k}$ from $g_1^{\nu'_k}$ for ciphertext challenges and $g_2^{\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})}$ from $g_2^{\nu'_k}$ for token challenges.

As is common in the generic bilinear group model [Sho97], we consider the challenger keeping record of all group elements the adversary has. It does so by keeping lists $P_{\mathbb{G},l}$ of linear polynomials in $\mathbb{Z}_p$ for each of the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. These polynomials use indeterminates for γ_i, $\alpha_i \pi(\beta_i, c_i)$, the $t_{\mathsf{ID}^{(j)}}$'s, $\alpha_i \pi(\beta_i, x_i^{(j)})$'s, $\alpha_i \pi(\beta_i, y^{(j)})$'s, $r_i^{(j)}$'s, and the $u_i^{(j)}$'s.

To simplify our reasoning, we will only look at polynomials $P_{\mathbb{G}_T,l}$ in $\mathbb{G}_T$. This is justified as we can transform any polynomial in $\mathbb{G}_1$ or $\mathbb{G}_2$ to a polynomial $P_{\mathbb{G}_T,l}$ in $\mathbb{G}_T$ through an additional query to the pairing oracle.

We can now say that the adversary wins the game if for a random assignment to all the indeterminates, any $P_{\mathbb{G}_T,i} \neq P_{\mathbb{G}_T,j}$ evaluates to the same value. We shall show that the adversary is not able to query for distinct polynomials $P_{\mathbb{G}_T,i}$, $P_{\mathbb{G}_T,j}$ such that, if the challenger plays the "real" experiment and if the indeterminates get assigned with random values, they will evaluate to the same value, except for negligible probability. Then, by the Schwartz lemma [Sch80] and the extended result of Shoup [Sho97], we can bound this probability of $P_{\mathbb{G}_T,i} \neq P_{\mathbb{G}_T,j}$ evaluating to the same value by $O(q^2 / p)$ if at most q group elements are given to the adversary.

In the case of a ciphertext challenge, we first have to bring the challenge response, which is an element of $\mathbb{G}_1$, to the target group $\mathbb{G}_T$. Since the adversary only has (linear combinations of) the elements g_2, $g_2^{u_i^{(j)}}$, $g_2^{\alpha_i \pi(\beta_i, y_i^{(j)}) u_i^{(j)}}$,

and $\prod_{i \in \mathcal{S}_\mathbf{y}} g_2^{u_i^{(j)} \gamma_i}$ in $\mathbb{G}_2$, it can only bring the challenge to $\mathbb{G}_T$ by pairing with one of these. Similarly, for token challenges, the adversary can only pair with the elements g_1, $g_1^{t_{\mathsf{ID}^{(j)}}}$, $g_1^{r_i^{(j)}}$, or $g_1^{\alpha_i \pi(\beta_i, x_i^{(j)}) r_i^{(j)} + t_{\mathsf{ID}^{(j)}} \gamma_i}$ in $\mathbb{G}_1$.

The resulting polynomials for these challenge responses are summarized in Table 4.1. Since the group elements are represented by uniformly independent values, the adversary can only distinguish between game k and game $k+1$ with more than a negligible advantage if it can construct at least one of the polynomials in this table.

Linear combinations. We now argue that the adversary cannot construct any of these challenges by looking at the components it has. We summarize the polynomials the adversary has access to, again by only looking at the elements in the target group $\mathbb{G}_T$, in Table 4.2. We have to show that no linear combination of the polynomials in Table 4.2 equals any of the polynomials in Table 4.1.

First, we look at the target queries for a ciphertext challenge to prove selective plaintext privacy and restricted adaptive predicate privacy without static corruptions. Later, we shall explain that the construction is indistinguishable if we allow for static corruptions as well.

Plaintext privacy. Observe that all polynomials for the ciphertext challenges contain $t_{\mathsf{ID}^{(c)}} \gamma_k$. This means that we only have to consider the forth column of Table 4.2 and the polynomial $t_{\mathsf{ID}} \sum_{i' \in \mathcal{S}_{\mathbf{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$. However, we require $t_{\mathsf{ID}^{(j)}} = t_{\mathsf{ID}^{(c)}}$, but we do not have any of the elements in the fourth column like this: If $t_{\mathsf{ID}^{(j)}} = t_{\mathsf{ID}^{(c)}}$, the adversary has requested an illegal query by using the challenge identifier $\mathsf{ID}^{(c)}$ in a query phase. Therefore, only the polynomial $t_{\mathsf{ID}^{(c)}} \sum_{i' \in \mathcal{S}_{\mathbf{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$ can be used, where the token query for $\mathbf{y}^{(j')}$ does not contain a wildcard in position k. This, in its turn, implies that the target ciphertext challenge has to contain exactly $t_{\mathsf{ID}^{(c)}} u_k^{(j')} \gamma_k$. Let $\mathbf{y}^{(\ell)}$ be an arbitrary queried vector such that $y_k^{(\ell)} \neq \star$. Now, by looking at Table 4.1, we conclude that we are left proving that no linear combination of the polynomials in Table 4.2 can form either $u_k^{(\ell)}(\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k)$ or $u_k^{(\ell)} \alpha_k \pi(\beta_k, y_k^{(\ell)})(\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k)$.

Table 4.1. Target polynomials in both indistinguishability games.

Ciphertext challenge

$\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k$	$u_i^{(j)} (\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k)$
$u_i^{(j)} \alpha_i \pi(\beta_i, y_i^{(j)}) (\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k)$	$(\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}^{(c)}} \gamma_k) \sum_{i \in \mathcal{S}_{\mathbf{y}^{(j)}}} u_i^{(j)} \gamma_i$

Token challenge

$\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$	$t_{\mathsf{ID}} \nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$
$r_i^{(j)} \nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$	$\left(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}} \gamma_i \right) \nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$

Table 4.2. Elements the adversary can query for in an indistinguishability game (up to linear combinations).

1	$t_{\mathsf{ID}^{(j)}}$	$r_i^{(j)}$	$r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i$
$u_{i'}^{(j')}$	$u_{i'}^{(j')} t_{\mathsf{ID}^{(j)}}$	$u_{i'}^{(j')} r_i^{(j)}$	$u_{i'}^{(j')} (r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i)$
$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')})$	$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) t_{\mathsf{ID}^{(j)}}$	$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) r_i^{(j)}$	$u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) (r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i)$
$\sum_{i' \in \mathcal{S}_{\mathbf{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$	$t_{\mathsf{ID}^{(j)}} \sum_{i' \in \mathcal{S}_{\mathbf{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$	$r_i^{(j)} \sum_{i' \in \mathcal{S}_{\mathbf{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$	$(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i) \sum_{i' \in \mathcal{S}_{\mathbf{y}^{(j')}}} u_{i'}^{(j')} \gamma_{i'}$

The polynomial $t_{\mathsf{ID}^{(c)}} u_k^{(\ell)} \gamma_k$ can be constructed using the expression $t_{\mathsf{ID}^{(c)}} \cdot \sum_{i \in \mathcal{S}_{\mathbf{y}^{(\ell)}} \setminus \{k\}} u_i^{(\ell)} \gamma_i$. Now, to cancel this summed term, we look at Table 4.2 and see that summing $u_{i'}^{(j')}(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i)$ for $j = c$, $j' = \ell$, and $i = i'$ is the only suitable polynomial. However, this introduces a new term $\sum_{i \in \mathcal{S}_{\mathbf{y}^{(\ell)}} \setminus \{k\}} u_i^{(\ell)} \nu_i \alpha_i \pi(\beta_i, x_i^{(c)})$. In Table 4.2 we only find $u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) \cdot r_i^{(j)}$ that can be used to cancel this new term. Setting $j = c$, $j' = \ell$, and $i = i'$, and summing results in

$$\sum_{i \in \mathcal{S}_{\mathbf{y}^{(\ell)}} \setminus \{k\}} u_i^{(\ell)} \alpha_i \pi(\beta_i, y_i^{(\ell)}) \nu_i.$$

Therefore, if the adversary queried for a token $\mathbf{y}^{(\ell)}$ where index k is not a wildcard and where

$$y_i^{(\ell)} = x_i^{(c)} \text{ for all } i \in \mathcal{S}_{\mathbf{y}^{(\ell)}} \setminus \{k\}, \tag{4.1}$$

it can cancel this most recent introduced term as well. Now, the adversary has constructed the polynomial $u_k^{(\ell)} t_{\mathsf{ID}^{(c)}} \gamma_k$ and so it is left to construct either $u_k^{(\ell)} \nu_k \alpha_k \pi(\beta_k, x_k^{(c)})$ or $u_k^{(\ell)} \alpha_k \pi(\beta_k, y_k^{(\ell)}) \nu_k \alpha_k \pi(\beta_k, x_k^{(c)})$. We claim that neither is possible.

By again looking at Table 4.2, we see that only $u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) r_i^{(j)}$, for $j = c$, $j' = \ell$, $i = i' = k$, and $y_k^{(\ell)} = x_k^{(c)}$, is suitable to construct the polynomial $u_k^{(\ell)} \nu_k \alpha_i \pi(\beta_i, x_k^{(c)})$. This means that we require both Equation (4.1) and $y_k^{(\ell)} = x_k^{(c)}$ to hold. However, this implies that $\mathsf{Match}^\star(\mathbf{x}^{(c)}, \mathbf{y}^{(\ell)}) = \textsc{true}$ and such a query for vector $\mathbf{y}^{(\ell)}$ is not allowed in the real-or-random game.

We try to construct the polynomial $u_k^{(\ell)} \alpha_k \pi(\beta_k, y_k^{(\ell)}) \nu_k \alpha_k \pi(\beta_k, x_k^{(c)})$ by looking at Table 4.2. The polynomial $u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')})(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i)$ for $j = c$, $j' = \ell$, $i = i' = k$ is the only suitable candidate to cancel the term. However, this introduces the new term $u_k^{(\ell)} \alpha_k \pi(\beta_k, y_k^{(\ell)}) t_{\mathsf{ID}^{(c)}} \gamma_k$, which can only be canceled by $u_k^{(\ell)} \alpha_k \pi(\beta_k, y_k^{(\ell)})(\nu_k \alpha_k \pi(\beta_k, x_k^{(c)}) + t_{\mathsf{ID}_{(c)}} \gamma_k)$ again. This reintroduces the original term that we wanted to construct. This means that we always have a term that we cannot cancel out, therefore we conclude that the adversary cannot construct this polynomial either.

Combining all results, we conclude that there is no linear combination and thus that the construction is selective plaintext private without static corruptions.

Restricted adaptive predicate privacy. We now look at the target queries for a token challenge. We can only construct $\nu_k \alpha_k \pi(\beta_k, y_k^{(c)})$ using the polynomial $u_k^{(j')} \alpha_k \pi(\beta_k, x_k^{(j)})$, where we require $j' = c$ and $x_k^{(j)} = y_k^{(c)}$. These

polynomials only occur in the last column of Table 4.2. However, the first cell in the column cannot be used as it does not contain indeterminates of the type u. The third and last cell, on the other hand, contain indeterminates that the adversary does not have access to, $\alpha_k \pi(\beta_k, y_k^{(c)})$, or can never cancel, $\gamma_i \gamma_{i'}$. Therefore, only the second cell for $i = i' = k$ and $j' = c$, $\nu_k(r_k^{(j)} \alpha_k \pi(\beta_k, x_k^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_k)$, seems to be a suitable candidate.

Let the ℓth query be an arbitrary ciphertext query where $x_k^{(\ell)} = y_k^{(c)}$, now if we use this polynomial, we introduce the new term $\nu_k t_{\mathsf{ID}^{(\ell)}} \gamma_k$. Looking at Table 4.2, we see that the only way to cancel this term is to use $t_{\mathsf{ID}^{(\ell)}} \sum_{i \in \mathcal{S}_{\mathbf{y}^{(c)}}} u_i^{(c)} \gamma_i$. The additionally introduced terms can be canceled with $u_{i'}^{(j')}(r_i^{(j)} \alpha_i \pi(\beta_i, x_i^{(j)}) + t_{\mathsf{ID}^{(j)}} \gamma_i)$ for $i = i'$, $j = \ell$, and $j' = c$. However, this again introduces the polynomial $\sum_{i \in \mathcal{S}_{\mathbf{y}^{(c)}} \setminus \{k\}} u_i^{(c)} r_i^{(\ell)} \alpha_i \pi(\beta_i, x_i^{(\ell)})$, which can only be canceled by $u_{i'}^{(j')} \alpha_{i'} \pi(\beta_{i'}, y_{i'}^{(j')}) r_i^{(j)}$ for $i = i'$ and $j' = \ell$, where we require $x_i^{(\ell)} = y_i^{(c)}$ for all $i \in \mathcal{S}_{\mathbf{y}^{(c)} \setminus \{k\}}$. Combining this requirement with the requirement we made earlier, that $x_k^{(\ell)} = y_k^{(c)}$, means that the adversary has to query for a ciphertext $\mathbf{x}^{(\ell)}$ such that $\mathsf{Match}^\star(\mathbf{x}^{(\ell)}, \mathbf{y}^{(c)}) = \textsc{true}$. Since this is not allowed in the game, we conclude that no token challenge can be made from a linear combination of the elements the adversary can query for.

Corruptions. We claim that, since we have proven the construction secure *without* corruptions and since the construction uses distinct, uncorrelated, encryptions keys usk_i for every client i, the construction is also secure under static corruptions. To see this, observe that the indistinguishability notion under static corruptions only guarantees indistinguishability of the uncorrupted vector components. This means that we can ignore the corrupted vector components and only require the adversary cannot learn the encryption key of client j if it learned the encryption key of client i. □

4.4.2 Chosen-Plaintext Security

The proposed construction is also chosen-plaintext secure as stated in Theorem 8. We remark that the proof does not rely on the use of random oracles.

Proof. We construct a challenger $\mathcal{B}$ capable of breaking the DDH assumption in $\mathbb{G}_1$ by using an adversary $\mathcal{A}$ that is able to win the chosen-plaintext with corruptions game with more than a negligible advantage.

We proof this though a series of hybrid games. Let game j be the game as defined in Definition 10, but where the first $j-1$ components of the challenge query are replaced by random elements. Note that game 1 is identical to the original game and that it is not possible for any adversary to gain an

advantage in game $n+1$. We are left to show that an adversary has at most a negligible advantage in distinguishing game j from game $j+1$.

Setup The challenger $\mathcal{B}$ receives the bilinear group parameters and the DDH instance $(A = g_1^a, B = g_1^b, Z) \in (\mathbb{G}_1)^3$. It chooses the hash function H and the encryption keys usk_i. It sets encryption key $\mathsf{usk}_j = (A, \beta_j \xleftarrow{R} \mathcal{K}, \gamma_j \xleftarrow{R} \mathbb{Z}_p^*)$ and chooses the rest of the encryption keys according to the scheme. The public parameters and the encryption keys usk_i are given to the adversary.

Challenge The adversary $\mathcal{A}$ submits an identifier ID^* and two vectors $\mathbf{x}_0^*$, $\mathbf{x}_1^*$ to the challenger. The challenger chooses $b \xleftarrow{R} \{0,1\}$ and sets $g_1^{r_j} = B$. Additionally, it picks values $r_i \xleftarrow{R} \mathbb{Z}_p^*$ for $1 \leq i \neq j \leq n$. It gives the challenge

$$\mathsf{ct}_i = \begin{cases} (H(\mathsf{ID}^*), g_1^{r_i}, R \xleftarrow{R} \mathbb{G}_1) & \text{if } i < j \\ (H(\mathsf{ID}^*), B, Z^{\pi(\beta_i, x_{b,i}^*)} H(\mathsf{ID}^*)^{\gamma_j}) & \text{if } i = j \\ (H(\mathsf{ID}^*), g_1^{r_i}, g_1^{\alpha_i r_i \pi(\beta_i, x_{b,i}^*)} H(\mathsf{ID}^*)^{\gamma_i}) & \text{if } i > j \end{cases}$$

for $1 \leq i \leq n$ to the adversary.

If the challenger is given $Z = g_1^{ab}$, then challenge ciphertext is identically distributed as the challenge ciphertext in game j and component j is a real encryption. If the challenger is given $Z \xleftarrow{R} \mathbb{G}_1$, then challenge ciphertext is identically distributed as the challenge ciphertext in game $j+1$ and component j is a random encryption.

Guess The challenger outputs its guess that $Z = g_1^{ab}$ if the adversary guesses that it is playing game j, and outputs its guess that $Z \xleftarrow{R} \mathbb{G}_1$ if the adversary guesses that it is playing game $j+1$.

If the adversary has a non-negligible advantage in distinguishing between game j and game $j+1$, the challenger obtains a non-negligible advantage in solving the DDH problem in group $\mathbb{G}_1$. □

4.5 Implementation and Evaluation

We have implemented a prototype of our construction with wildcards to get a better understanding of its performance. The implementation[3] uses the pairing-based cryptography (PBC) library[4] that allows one to easily change the underlying curve and its parameters.

[3] https://github.com/CRIPTIM/multi-client-monitoring
[4] https://crypto.stanford.edu/pbc/

Instantiating the Pseudorandom Permutation Our construction uses a PRP π to permute an element in $\mathbb{Z}_p$. However, since we use the outcome of the permutation to exponentiate a generator in $\mathbb{G}_1$ and $\mathbb{G}_2$, we can instead directly map values in $\mathbb{Z}_p$ to one of these groups respectively. The pseudorandom function (PRF) proposed by Naor and Reingold [NRO4] exactly achieves this. Their PRF maps a message $x \in \mathcal{M} \subseteq \{0, \ldots, 2^m - 1\} \subseteq \mathbb{Z}_p$ using a key $\mathbf{b} = \{ b_i \xleftarrow{R} \mathbb{Z}_p^* \mid 0 \leq i \leq m \}$ to an element in a group $\langle g \rangle$ of prime order p. The PRF F is defined as

$$F(\mathbf{b}, x) = g^{b_0 \prod_{i=1}^{m} b_i^{x[i]}},$$

where $x[i] \in \{0, 1\}$ denotes the ith bit of message x. The advantage of using this PRF over a PRP is that it is relatively simple to compute while it is provably secure under the DDH assumption.

We apply the PRF to both the Encrypt and the KeyGen* algorithms to obtain ciphertexts of the form

$$\mathsf{ct}_i = \left(H(\mathsf{ID}), g_1^{r_i}, g_1^{\alpha_i \prod_{j=1}^{m} \beta_{i,j}^{x_i[j]} r_i} H(\mathsf{ID})^{\gamma_i} \right),$$

and tokens of the form

$$\mathsf{esk}_{\mathbf{y}} = \left(\left\{ g_2^{u_i}, g_2^{\alpha_i \prod_{j=1}^{m} \beta_{i,j}^{y_i[j] u_i}} \mid i \in S_{\mathbf{y}} \right\}, \prod_{i \in S_{\mathbf{y}}} (g_2^{\gamma_i})^{u_i} \right).$$

Notice that we use $b_0 = \alpha_i$ and $b_j = \beta_{i,j}$. In addition, observe that it is not necessary to know the value α_i to compute a ciphertext or token, as long the value $g_1^{\alpha_i}$, or $g_2^{\alpha_i}$ respectively, is known.

Performance Measurements We ran several performance evaluations on a notebook containing an Intel Core i5-4210U@1.7 GHz CPU, running Debian GNU/Linux. We chose to evaluate the system using an MNT curve [MNTO1] over a 159 bit base field size with embedding degree 6.

As expected from the theoretical performance analysis in Section 4.3.4, both the KeyGen* and Eval* algorithms scale linearly in the number of non-wildcard components used. The KeyGen* algorithm spends, on average, 19 ms to encrypt a non-wildcard component. To evaluate a token that contains no wildcards using n ciphertexts, takes $4.5n + 10$ ms on average. The Setup algorithm scales linearly as well, spending on average 18 ms per client to create their public and private keys. The Encrypt algorithm is the fastest, taking only 2.6 ms for an individual client to encrypt a message $x_i \in \{0, \ldots, 15\}$.

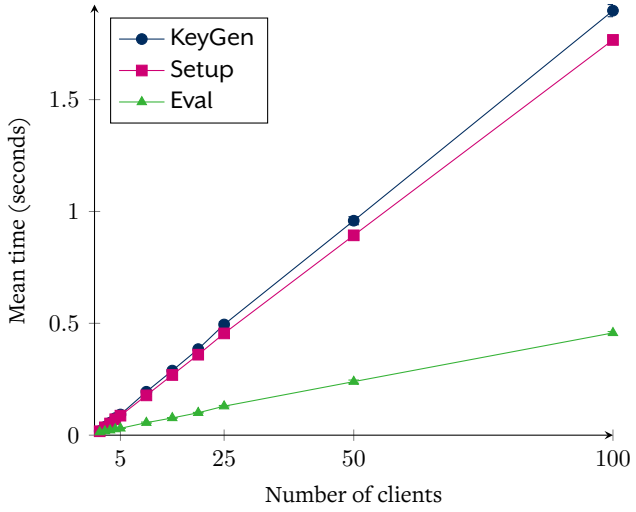


Figure 4.2. Performance measurements of the implementation using an MNT-159 curve.

In Figure 4.2 the average computational time is plotted against the number of clients involved in the computation. No wildcards were used in the KeyGen⋆ and Eval⋆ algorithms to obtain these timing results, meaning that the algorithms are identical to KeyGen and Eval, respectively.

Considering an example of the monitoring of several critical infrastructure operators, we remark that a typical information-sharing community (*e.g.*, an ISAC) consists of about 10 parties. So, if every party sends 5 distinct messages for each identifier (*e.g.*, every party has five subsystems to be monitored), we would require a system of about 50 clients. We see that in such a realistically sized system we can evaluate about 250 predicates per minute. Optimizations such as the preprocessing of pairings can increase the number of predicate evaluations per minute.

4.6 Conclusion

By designing a special-purpose multi-client functional encryption scheme, it is possible to create a practical privacy-preserving monitoring system. To achieve this, we defined multi-client predicate-only encryption (MC-POE) and corresponding security definitions for the protection of both the messages of the individual clients and the predicates. Our proposed construction for such an MC-POE scheme is capable of conjunctive equality testing over vector components which can include wildcards. The performance evaluation of our implementation shows that the evaluation time of a predicate scales linearly in the number of clients, where a predicate defined over 20 clients can be evaluated in a tenth of a second. Additionally, we see that the encryption algo-

rithm is very lightweight, making it suitable to run on resource-constrained devices.

Future work will include the construction of an MC-POE scheme which will allow for more expressive functionality, while remaining efficient enough to run in practice and keeping the confidentiality of both the messages and the predicates. Additionally, further research is needed to construct an MC-POE scheme that is fully secure in the standard model.

5 General Predicates

Multi-authority Predicate Encryption

We construct the first multi-client predicate-only encryption for equality testing in the previous chapter. In this chapter, we continue along the same lines with functional encryption for various predicate classes. We propose the first generic construction for fully secure decentralized multi-authority predicate encryption. In a multi-authority predicate encryption scheme, ciphertexts are associated with one or more predicates from various authorities and only if a user has a set of decryption keys that evaluates all predicates to TRUE, the user is able to recover the message. In a decentralized system, anyone can create a new authority and issue decryption keys for their own predicates. We introduce the concept of a *multi-authority admissible pair encoding scheme*, and based on these encodings, we give a generic conversion algorithm that allows us to easily combine various predicate encryption schemes into a multi-authority predicate encryption variant. The resulting encryption schemes are proven *fully secure* under standard subgroup decision assumptions in the random oracle model. Finally, by instantiating several concrete multi-authority admissible pair encoding schemes and applying our conversion algorithm, we are able to create a variety of novel multi-authority predicate encryption schemes.

This chapter is based on the work "A Multi-authority Approach to Various Predicate Encryption Types" [KPJ20], published in *Designs, Codes and Cryptography* (DESI). While this chapter treats the proposed construction as multi-authority predicate encryption, we explain in Section 5.1.1 that this concept is a type of multi-client functional encryption. To answer Research Question 1, we view the construction as a multi-client functional encryption scheme.

5.1 Introduction

Predicate encryption (PE) is a type of public-key encryption, where the out-

come of decryption is controlled by a relation R. A user possessing a decryption key associated with value y, is only able to recover the plaintext of a ciphertext associated with value x, if the relation $R(x, y)$ holds. Many different types of PE have been proposed, each characterizable by the family of relations they support. Examples of PE types include identity-based encryption (IBE) [BF01] (where the relation is equality testing), attribute-based encryption (ABE) [SW05] (equality testing joined with logical AND and OR gates), hidden vector encryption (HVE) [BW07] (vector equality testing with wildcard support), and inner-product predicate encryption (IPPE) [KSW08] (testing whether two vectors are orthogonal). Even more advanced schemes, such as schemes capable of evaluating relations based on regular languages, exist as well [Wat12].

A drawback of standard PE is that a single party, the *authority*, is responsible for creating the decryption keys for all users in the system. As a direct consequence, this authority can decrypt all messages since the authority has to be able to create every possible decryption key. Thus, relying on a single authority has not only consequences for the scalability of the system, but also for the trust relations. In natural situations, we would rather appoint multiple authorities, where each authority is responsible for issuing keys in their own realm. For example, when handling data from a clinical trial, we demand that only medical doctors affiliated to a research institute have access to the data. A hospital could then be responsible for issuing a decryption key for "medical doctor," while a university would be responsible for issuing the decryption key for "researcher."

The question whether it is possible to construct such a multi-authority scheme was first raised by Sahai and Waters [SW05]. In a multi-authority predicate encryption (MA-PE) scheme, ciphertexts are associated with one or more predicates from various authorities. Users are then only able to decrypt the ciphertext if their keys make all predicates associated with the ciphertext evaluate to TRUE. The first proposed MA-PE constructions [Cha07; CC09; MKE09] either require interaction between all authorities, or solely address the scalability problem and still require a master secret which can be used to decrypt all messages. To address both problems at the same time, Lewko and Waters [LW11] proposed a *decentralized* scheme. However, a limitation of all previous proposed MA-PE constructions, is that they only address the special case of multi-authority attribute-based encryption (MA-ABE), rather than the more general MA-PE.

We propose the first generic framework for creating decentralized multi-authority predicate encryption. Our framework supports several predicate types, such as multi-authority IBE, multi-authority ABE, and multi-authority

IPPE. We also provide an instantiation for each of these predicate families. Since our solution is decentralized, we address both the trust and scalability issues: No party is required to hold a global master secret and new authorities can be created without requiring any form of interaction. Lastly, we prove that the encryption schemes resulting from our framework are *fully secure*.

Our construction for an MA-PE scheme can be seen as the combination of multiple parallel instantiations of a (modified) single authority PE scheme with a "multi-authority layer" on top. Basically, the MA-PE scheme first fixes the group parameters and every new authority can than instantiate a new PE scheme in this group. To encrypt a message, a user blinds the message with a random number and split this random number using additive secret sharing into various shares. Next, each of the shares are encrypted using the PE scheme's public key. Decryption works by first decrypting all shares to recover the random number and then unblind the blinded message. However, described as such, the scheme would be vulnerable to a collusion attack, *i.e.*, users combining their knowledge to gain access to messages they should not have access to. To see this, assume we have a ciphertext that may only be decrypted by students older than 21. Now, two colluding users, one with the "student" attribute and another one with the "over-21" attribute, can each obtain part of the shares. If they combine their shares they are able to unblind the blinded message, while neither of them should have been able to. To prevent this attack, we make sure that during the decryption of a share, randomness specific to the user is added. Only if the shares of the same user are combined, this user specific randomness cancels out.

To support a variety of PE schemes for the use in a decentralized MA-PE scheme, we introduce the concept of multi-authority admissible pair encoding schemes (MA-PESS). An MA-PES can be "compiled" into a PE scheme compatible with an MA-PE scheme using our conversion algorithm. The definition of an MA-PES is an extended variant of the recently introduced concept of pair encoding schemes (PESS) [Att14; AC16; AC17]. Such a (multi-authority admissible) pair encoding scheme describes how a predicate can be encoded in an encryption scheme, without having to consider the group structure the scheme is instantiated in. This separation of encoding and group structure greatly simplifies the construction of new (multi-authority) PE schemes since it is relatively easy to prove an MA-PES secure compared to proving the entire encryption scheme secure. After proving the MA-PES secure, we can simply apply our conversion algorithm to turn the secure MA-PES into a secure MA-PE scheme.

Using the proposed conversion algorithm, we are able to combine various PE schemes for different predicates (*e.g.*, IBE, ABE, or IPPE) into an MA-PE

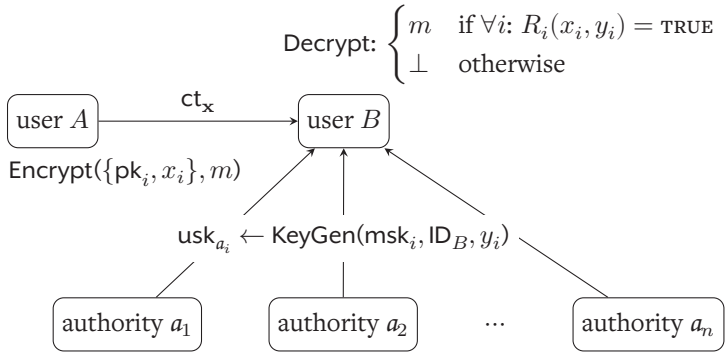
scheme using AND gates between the predicates. While the need for OR gates can be circumvented by writing the global policy in disjunctive normal form (DNF) and encrypting the plaintext for each of the conjunctive clauses, we could also directly support OR gates by slightly chaning the algorithm: By using Shamir secret sharing (SSS) instead of additive secret sharing, policies can also contain OR gates [LW11].

We prove that applying our conversion algorithm on a secure MA-PES results in a *fully secure* MA-PE scheme in the random oracle model. In our *full security* game for multiple authorities, several authorities may be corrupted while the adversary may query the challenger for both the creation of new authorities and for decryption keys of its choice. We use a variant of the *dual system encryption technique* to prove our construction secure. The dual system proof technique, first introduced in the seminal work by Waters [Wat09] and later refined by a series of subsequent work [LOS+10; LW10; LW12; CW13], uses *semi-functional* ciphertexts and keys in the proofs. A semi-functional ciphertext can be decrypted using a normal key, and a normal ciphertext can be decrypted by a semi-functional key (of course, in both cases we still require that the relation R holds). However, a semi-functional ciphertext can never be decrypted by a semi-functional key, not even if the relation R holds. To prove a scheme secure, we use a series of hybrid games. In the final game, the adversary receives a semi-functional challenge ciphertext and only semi-functional keys, meaning that the adversary has no chance in correctly decrypting the challenge ciphertext, and thus making it impossible for the adversary to gain a non-negligible advantage in winning the game.

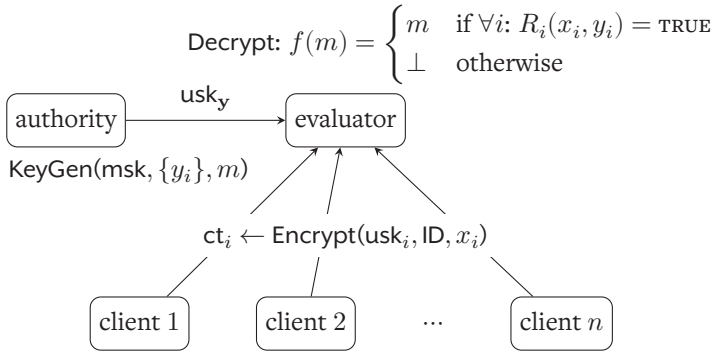
5.1.1 Relation to Multi-client Functional Encryption

While in this chapter, we primarily discuss the concept of multi-*authority* encryption, we note that this is not much different from multi-*client* encryption. The relation between MA-PE and multi-client functional encryption (MC-FE) for predicates becomes apparent if we interchange the Encrypt and KeyGen algorithms. We schematically depict the settings in which MA-PE and MC-FE are typically used in Figure 5.1.

While in an MA-PE scheme, KeyGen is used by one of several authorities to generate a key specifically for user with identity ID, in an MC-FE scheme, Encrypt is used by one of several clients to generate a ciphertext specifically for identifier ID. In MA-PE, this ID is needed to prevent user collusion, while in MC-FE the ID similarly prevents mix-and-match attacks. Thus, MC-FE for predicates can be seen as an MA-PE scheme where the clients/authorities create for every session identifier ID a new message associated with y_i for a user with identity ID. Of course, in MC-FE, it is crucial that the ciphertexts



(a) Typical setting of multi-authority predicate encryption.



(b) Typical setting of multi-client functional encryption for predicates.

Figure 5.1. The relation between MA-PE and MC-FE. We see from the two figures that the concepts are almost identical. Just the parties, algorithms, and transmitted data are termed differently. One of the more significant differences is that in an MA-PE scheme the Encrypt algorithm is public key, while the in an MC-FE scheme the corresponding KeyGen algorithm requires knowledge of the msk.

hide the plaintext messages x_i, while in the MA-PE setting the corresponding KeyGen algorithm does not necessarily needs to hide the values y_i. However, PE schemes satisfying the predicate-privacy notion also hide these values y_i. A direct consequence of a PE scheme satisfying the predicate-privacy notion, is that such a scheme has to be secret key, instead of public key [SSW09]. This immediately explains why in MC-FE the KeyGen algorithm requires knowledge of the master secret msk, while the corresponding algorithm in MA-PE, Encrypt, can in fact be public key.

It might seem strange at first that in MC-FE for predicates the KeyGen algorithm also takes a message m to encrypt. However, this is very natural in a monitoring system as described in Section 4.1.1: We want a monitor to only learn more information (*i.e.*, recover a message) about an incident (encoded in the relation R) if the incident actually occurs. Besides the message m, the values y_i can also be hidden. Similar to the predicate-privacy notion, the plaintext-privacy notion guarantees that the KeyGen algorithm in an MC-FE scheme hides the values y_i. In Chapter 4, we construct a scheme that satisfies

both the predicate-privacy and the plaintext-privacy notion at the same time.

5.2 Preliminaries

We often work with vectors of group elements $(g^{v_1}, \ldots, g^{v_n})$, written as $g^{\mathbf{v}}$. We use the notation for a predicate family by Attrapadung [Att14]. Let $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$, for some constant $c \in \mathbb{N}$, denote the predicate family for relations $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{\text{TRUE}, \text{FALSE}\}$. Here, a relation is equivalent to a predicate function where $\mathcal{X}_\kappa$, the ciphertext attribute space, and $\mathcal{Y}_\kappa$, the key attribute space, are mapped to a TRUE/FALSE output. A predicate P_κ can be described by its family index κ. We often use $\kappa(a)$ to denote that the index is specific to an authority a.

Our construction uses a Type 1 pairing (see Definition 4) of *composite order* $N = p_1 p_2 p_3$ for distinct primes p_1, p_2, and p_3. We use the function $\mathcal{G}_1(1^\lambda)$ to generate the parameters for a composite-order bilinear map for security parameter λ.

5.2.1 Complexity Assumptions

The security of our construction relies on several instances of the family of the General Subgroup Decision Assumption [BWY11]. These assumptions are identical to the assumptions used by the MA-ABE scheme of Lewko and Waters [LW11].

Assumption 4. Let the group parameters $\text{GP} = (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e, g)$ be generated by $\mathcal{G}_1(1^\lambda)$ and $g_1 \xleftarrow{R} \mathbb{G}_1$. Given g_1, it is hard to distinguish $\widehat{h} \xleftarrow{R} \mathbb{G}$ from $\widehat{h}_1 \xleftarrow{R} \mathbb{G}_1$. That is, the advantage of any probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ in distinguishing,

$$\Big|\Pr[\mathcal{A}((\text{GP}, g_1), \widehat{h}) = 1] - \Pr[\mathcal{A}((\text{GP}, g_1), \widehat{h}_1) = 1]\Big|,$$

is negligible in the security parameter λ.

Assumption 5. Let the group parameters $\text{GP} = (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e, g)$ be generated by $\mathcal{G}_1(1^\lambda)$, and $g_1, h_1, \widehat{h}_1 \xleftarrow{R} \mathbb{G}_1$, $h_2, \widehat{h}_2 \xleftarrow{R} \mathbb{G}_2$, and $g_3 \xleftarrow{R} \mathbb{G}_3$. Given g_1, $h_1 h_2$, and g_3, it is hard to distinguish $\widehat{h}_1$ from $\widehat{h}_1 \widehat{h}_2$. That is, the advantage of any p.p.t. adversary $\mathcal{A}$ in distinguishing,

$$\Big|\Pr[\mathcal{A}((\text{GP}, g_1, h_1 h_2, g_3), \widehat{h}_1) = 1] - \Pr[\mathcal{A}((\text{GP}, g_1, h_1 h_2, g_3), \widehat{h}_1 \widehat{h}_2) = 1]\Big|,$$

is negligible in the security parameter λ.

Assumption 6. Let the group parameters $\text{GP} = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e, g)$ be generated by $\mathcal{G}_1(1^\lambda)$, and pick elements $g_1, h_1, \hat{h}_1 \xleftarrow{R} \mathbb{G}_1$, $h_2', \hat{h}_2 \xleftarrow{R} \mathbb{G}_2$, and $h_3, h_3', \hat{h}_3 \xleftarrow{R} \mathbb{G}_3$. Given g_1, h_1h_3, and $h_2'h_3'$, it is hard to distinguish $\hat{h}_1\hat{h}_2$ from $\hat{h}_1\hat{h}_3$. That is, the advantage of any p.p.t. adversary $\mathcal{A}$ in distinguishing,

$$\Big|\Pr[\mathcal{A}((\text{GP}, g_1, h_1h_3, h_2'h_3'), \hat{h}_1\hat{h}_2) = 1] - \Pr[\mathcal{A}((\text{GP}, g_1, h_1h_3, h_2'h_3'), \hat{h}_1\hat{h}_3) = 1]\Big|,$$

is negligible in the security parameter λ.

Assumption 7. Let the group parameters $\text{GP} = (N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e, g)$ be generated by $\mathcal{G}_1(1^\lambda)$, and pick $g_1 \xleftarrow{R} \mathbb{G}_1$, $g_2 \xleftarrow{R} \mathbb{G}_2$, $g_3 \xleftarrow{R} \mathbb{G}_3$, and $a, b, c, d, \xi \xleftarrow{R} \mathbb{Z}_N$. Given g_1, g_2, g_3, g_1^a, $(g_1g_3)^b$, g_1^c, and $g_1^{ac}g_3^d$, it is hard to distinguish $e(g_1, g_1)^{abc}$ from $e(g, g)^\xi$. That is, the advantage of any p.p.t. adversary $\mathcal{A}$ in distinguishing,

$$\Big|\Pr[\mathcal{A}((\text{GP}, g_1, g_2, g_3, g_1^a, (g_1g_3)^b, g_1^c, g_1^{ac}g_3^d), e(g_1, g_1)^{abc}) = 1] - \Pr[\mathcal{A}((\text{GP}, g_1, g_2, g_3, g_1^a, (g_1g_3)^b, g_1^c, g_1^{ac}g_3^d), e(g, g)^\xi) = 1]\Big|,$$

is negligible in the security parameter λ.

5.2.2 Multi-authority Predicate Encryption

A decentralized multi-authority predicate encryption (MA-PE) scheme differs from a single authority PE scheme in several key aspects. Most importantly, any party can use the global public parameters to create a new authority a. Using these global parameters, it creates its own public/private key pair for a predicate indexed by $\kappa(a)$.

Furthermore, since every authority has its own public key, the encryption algorithm requires one or more public keys as input. Naturally, only the public keys of the authorities $\mathcal{A}$ involved in the access policy are required to encrypt a message. Besides the public keys, the algorithm also requires the ciphertext values x_a for each of the authorities $a \in \mathcal{A}$. Note that these values may come from distinct domains, as this value space $\mathcal{X}_{\kappa(a)}$ depends on the predicate index $\kappa(a)$.

Finally, to prevent user collusion, every user in the system get its own globally unique identity ID from an identity space $\mathcal{ID}$. Decryption keys are issued to a specific user and are bound to their personal ID. This prevents collusion attacks in which distinct users try to combine their key to decrypt a ciphertext that may only be decrypted by users that possess *all* required keys themselves.

Definition 11 (Multi-authority Predicate Encryption)**.** A decentralized multi-authority predicate encryption (MA-PE) scheme is a collection of the following five probabilistic polynomial time algorithms.

GlobalSetup$(1^\lambda) \to \mathsf{pp}$**.** On input of the security parameter λ, the algorithm outputs the global public parameters pp of the scheme. The output of GlobalSetup additionally defines the message space $\mathcal{M}$, the identity space $\mathcal{ID}$, and a number $N \in \mathbb{N}$ (these may be implicitly defined by pp).

All of the following algorithms (implicitly) use the global public parameters pp.

AuthoritySetup$(\mathsf{pp}, \mathsf{par}_a) \to (\mathsf{pk}_a, \mathsf{msk}_a)$**.** On input of the public parameters pp and some additional parameters par_a (describing the predicate $P_{\kappa(a)}$), the algorithm outputs a public key pk_a and an authority secret key msk_a for authority a. The algorithm AuthoritySetup (implicitly) sets $\kappa(a)$ to the tuple (N, par_a).

Encrypt$(\{(\mathsf{pk}_a, x_a)\}_{a\in\mathcal{A}}, m) \to \mathsf{ct}$**.** The algorithm Encrypt takes a set of public keys $\{\mathsf{pk}_a\}$ from authorities $a \in \mathcal{A}$, values $\{x_a \in \mathcal{X}_{\kappa(a)}\}_{a\in\mathcal{A}}$, and a message $m \in \mathcal{M}$ as input and outputs a ciphertext ct.

KeyGen$(\mathsf{msk}_a, y, \mathsf{ID}) \to \mathsf{usk}_{y,\mathsf{ID}}$**.** The algorithm KeyGen takes an authority secret key msk_a of authority a, a value $y \in \mathcal{Y}_{\kappa(a)}$, and an identity $\mathsf{ID} \in \mathcal{ID}$ as input and outputs a user secret key $\mathsf{usk}_{y,\mathsf{ID}}$.

Decrypt$(\{\mathsf{usk}_{y,\mathsf{ID}}\}_y, \mathsf{ct}) \to \{m, \perp\}$**.** On input of user secret keys $\{\mathsf{usk}_{y,\mathsf{ID}}\}$, all issued to the same identity ID, and a ciphertext ct, the algorithm outputs either a message m or the distinctive symbol $\perp$.

Correctness Correctness is defined such that if all predicates $P_{\kappa(a)}$ can be evaluated to TRUE, the ciphertext can be decrypted with an overwhelming probability. That is, an MA-PE scheme is *correct* if for any combination of ciphertext ct, created using Encrypt with any message $m \in \mathcal{M}$ and values $\{x_a \in \mathcal{X}_{\kappa(a)}\}_{a\in\mathcal{A}}$, together with keys for the authorities a specified in the ciphertext ct, $\{\mathsf{usk}_{y_a,\mathsf{ID}}\}_{a\in\mathcal{A}}$ for any identity $\mathsf{ID} \in \mathcal{ID}$, $P_{\kappa(a)}(x_a, y_a) = \text{TRUE}$, then

$$\Pr\left[\mathsf{Decrypt}(\{\mathsf{usk}_{y_a,\mathsf{ID}}\}, \mathsf{ct}) \neq m\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the coins of GlobalSetup, AuthoritySetup, Encrypt, and KeyGen.

5.2.3 Multi-authority Predicate Encryption Security

We define security in terms of an indistinguishability game where the adversary may query for several decryption keys and has to decide on the message encrypted in the challenge ciphertext. The adversary may also query for the creation of new authorities and also statically corrupt new authorities. The static corruption of an authority is modeled by letting the adversary create a public/private key pair for a new authority. The adversary may then request the challenger to encrypt the challenge message using the public keys of uncorrupted and corrupted authorities. Note that this implies a static corruption model similar to [LW11], as none of the authorities associated with the challenge ciphertext may be corrupted after the challenge phase. The difference is that we do not require all authorities to be specified during Setup, but allow for "Authority Setup" queries.

Definition 12 (Full Security of MA-PE)**.** A multi-authority predicate encryption scheme is *fully secure* if any p.p.t. adversary $\mathcal{A}$ has at most a negligible advantage in winning the following game.

Setup The GlobalSetup algorithm is run and the challenger creates an empty set I to hold the uncorrupted authorities in the system.

Query 1 The adversary may query the challenger for two types of queries. Additionally, it can also create new authorities using the global parameters, *i.e.*, without needing to query the challenger.

- **Authority Setup** The adversary queries for a new authority by sending the parameters par_a (describing a predicate) to the challenger. The challenger runs AuthoritySetup using par_a and gives the resulting public key pk_a to the adversary. Additionally, it adds a to the set of uncorrupted authorities I.

- **User Secret Key** By sending a tuple $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$, where $a \in I$, to the challenger, the adversary requests the user secret key $\mathsf{usk}_{y,\mathsf{ID}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}_a, y, \mathsf{ID})$ from the challenger. If the challenger has received a key request for the combination (a, ID) before, it aborts the game.[1] Otherwise, it returns the user secret key $\mathsf{usk}_{y,\mathsf{ID}}$.

Challenge The adversary sends a tuple $(m_0, m_1, \{x_a^*\}_{a \in \mathcal{A}^*})$ to the challenger, where $\mathcal{A}^*$ is a set of authorities chosen by the adversary. For each authority $a \in \mathcal{A}^*$ the adversary created itself, it also sends the public key pk_a to

[1] The construction of Lewko and Waters [LW11] also requires that no authority may issue a key to the same user twice, although they do not make this requirement explicit.

the challenger. We denote these authorities created by the adversary by the set $\bar{I} = \mathcal{A}^* \setminus I$.

For each ID that was used in a key query, the challenger checks if there exists an uncorrupted authority $a' \in \mathcal{A}^* \cap I$, such that either *no* user secret key query $(a', y_{a'}, \mathsf{ID})$ has been made, or $P_{\kappa(a')}(x^*_{a'}, y_{a'}) = \text{FALSE}$ for the queried $(a', y_{a'}, \mathsf{ID})$. If so, it chooses a bit $b \xleftarrow{R} \{0,1\}$ and returns the challenge $\mathsf{Encrypt}(\{\mathsf{pk}_a\}_{a\in\mathcal{A}^*}, \{x^*_a\}_{a\in\mathcal{A}^*}, m_b)$. Otherwise, the challenger aborts the game.

Query 2 Same as Query 1, with the additional restriction that new key queries must not violate the constraint described in Challenge.

Guess The adversary makes a guess b' for bit b. We define the advantage of the adversary in winning the game as

$$\Pr[b' = b] - \frac{1}{2}.$$

5.3 Related Work

Up until now, the vast majority of MA-PE schemes proposed in literature are MA-ABE schemes. The first MA-ABE schemes either require the introduction of a central party that is even able to decrypt all ciphertexts [Cha07; MKE09] or do not allow for the addition of new authorities once the system is set up [CC09]. The first practical MA-ABE scheme came with the introduction of *decentralized* MA-ABE [LW11]. A decentralized MA-PE scheme does not require any central party and anyone can start a new authority completely independent of all other parties. However, the current decentralized MA-ABE schemes [LW11; OT13; RW15] only support a single fixed construction and lack the ability to be used with any predicate family other than ABE. Moreover, in our construction, each authority can choose its own predicate family, which allows for the combination of several predicate systems, *e.g.*, we can combine ABE and IPPE in a single MA-PE scheme.

In 2014, both Wee [Wee14] and Attrapadung [Att14] observed that many of the schemes proven secure under the dual system encryption technique could be split into an encoding of the predicate and the group structure this encoding is instantiated in. Three variants of these encodings exist: predicate encoding [Wee14], pair encoding [Att14], and the later introduced tag-based encoding [KSG+16]. Several newer works build on various improvements of the concepts of predicate encodings [CGW15; ABS17] and pair encodings [AC16; AC17; ABS17]. Because pair encodings are the most general of the three, we base our work on pair encodings. For the instantiation of the group

structure, composite-order and prime-order groups can be used [CW13; CW14; AC16]. In this work, we instantiate our decentralized MA-PE scheme in a composite-order group setting, resulting in the first generic MA-PE scheme. The previously proposed prime-order group structure cannot be directly used, since our construction uses a system based on three subgroups, instead of the more common two subgroups.

The MA-PE schemes resulting from our conversion algorithm are fully secure, similar to notions used before [LW11; OT13]. Our notion is slightly more permissive in the sense that not all authorities need to be announced at the start of the game, but the adversary can query for new authorities throughout the game. Weaker security notions, *e.g.*, selective or static security games [RW15], or the use of the generic group model often allow for simpler and more efficient constructions at the costs of security.

A special use of our MA-PE construction is the combination of various predicate families into a single authority PE scheme, *i.e.*, the (single) authority creates multiple key pairs, each for a distinct predicate family. Constructions of these combined PE schemes was first studied for the combination of ciphertext-policy attribute-based encryption (CP-ABE) with key-policy attribute-based encryption (KP-ABE) [AI09; AY15]. Recently, Ambrona, Barthe, and Schmidt [ABS17] give generic transformations to combine arbitrary predicate encodings into a new (single authority) predicate encoding scheme. Their approach differs from ours, since we do not *transform* encodings into an encoding for a combined predicate, but *convert* our encodings into an encryption scheme for combined predicates.

We stress that our achieved functionality of decentralized multi-authority inner-product predicate encryption (MA-IPPE) is different from the works on multi-input inner product encryption (MI-IPE) [AGR+17; DOT18]. In inner product encryption, the decryption algorithm outputs the inner product of two encrypted vectors, while in IPPE, the orthogonality of two vectors determines whether an encrypted message can be decrypted. The work by Michalevsky and Joye [MJ18] achieves a specific form of MA-IPPE under a notion of decentralization that requires a semi-honest authority and coordination among the authorities during key generation. Their paper brings up the challenge to realize what the authors call "full decentralization" which we tackle in this work. Moreover, our construction achieves this type of "full" decentralization for various MA-PE types, including MA-IPPE.

5.4 Multi-authority Admissible Pair Encoding Scheme

We extend the definition of a pair encoding [Att14; AC17] to a multi-authority setting. A multi-authority admissible pair encoding scheme (MA-PES) is defined for a *single* authority a. We shall later show how we can convert *several* MA-PESs into a *single* multi-authority predicate encryption (MA-PE) scheme.

We choose to extend the definition of a pair encoding scheme (PES) as defined by Agrawal and Chase [AC17] since it is well-structured—although it may be a bit difficult to grasp at first. To get a better understanding of the scheme, it is convenient to think of the encodings as the variables in the exponents in the encryption scheme. The values $\mathbf{b}$ correspond to an authority's public key, while $\mathbf{s}, \hat{\mathbf{s}}$ and $\mathbf{r}, \hat{\mathbf{r}}$ correspond to the randomness used in the encryption and key generation algorithms, respectively. The algorithms EncCt and EncKey encode the ciphertext value x and key value y, respectively, by returning one or more multivariate polynomials of a restricted form. The variables $b_1, \ldots, b_n$ can occur in both the ciphertext and the key encoding, so they are termed *common*. These common variables may be multiplied with *non-lone* a variable s_i (in a ciphertext encoding) or r_i (in a key encoding). A *lone* variable, indicated by a hat, *e.g.*, $\hat{r}_i$, is never multiplied with a common variable, but may be added as an independent term to the polynomial. Two special variables, α in the key encodings—corresponding to the authority's secret key—and ω in the ciphertext encodings, are always present in at least one of the polynomials. Basically, the encodings of a ciphertext contain linear combinations of monomials ω, $\hat{s}_i$, and $s_i b_j$, while key encodings contain linear combinations of α, $\hat{r}_i$, and $r_i b_j$.

Recall that our construction can be understood as a combination of several *multi-authority admissible* PE schemes using a "multi-authority layer" that withstands collusion attacks. During the decryption of such a multi-authority admissible PE scheme, randomness specific to the user is added to prevent collusion attacks. In our MA-PES, this randomness is represented in the correctness requirement by the newly added term ωr_0, where r_0 corresponds to the user's ID.

Our changes with respect to the PES definition by Agrawal and Chase [AC17] are highlighted in red.

Definition 13 (Multi-authority Admissible Pair Encoding Scheme)**.** A multi-authority admissible pair encoding scheme (MA-PES) for a predicate function $P_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{\text{FALSE}, \text{TRUE}\}$ indexed by $\kappa = (N, \mathsf{par})$, where par specifies some parameters, is given by the following four *deterministic* polynomial-time algorithms.

AuthorityParam$(\mathsf{par}) \to n$**.** When given par as input, AuthorityParam out-

puts $n \in \mathbb{N}$ that specifies the number of common variables, which we denote by $\mathbf{b} = (b_1, \ldots, b_n)$.

EncCt$(N, x) \to (w_1, w_2, \mathbf{c}(\omega, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}))$**.** On input $N \in \mathbb{N}$ and $x \in \mathcal{X}_{(N,\mathsf{par})}$, EncCt outputs a vector of polynomials $\mathbf{c} = (c_1, \ldots, c_{w_3})$ in non-lone variables $\mathbf{s} = (s_0, s_1, \ldots, s_{w_1})$ and lone variables ω and $\hat{\mathbf{s}} = (\hat{s}_1, \ldots, \hat{s}_{w_2})$. For $\ell \in [w_3]$, where $\eta_\ell, \eta_{\ell,z}, \eta_{\ell,i,j} \in \mathbb{Z}_N$, the ℓth polynomial is given by

$$c_\ell(\omega, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}) = \eta_\ell \omega + \sum_{z \in [w_2]} \eta_{\ell,z} \hat{s}_z + \sum_{i \in [w_1]^+} \sum_{j \in [n]} \eta_{\ell,i,j} s_i b_j.$$

EncKey$(N, y) \to (m_1, m_2, \mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}))$**.** On input $N \in \mathbb{N}$ and $y \in \mathcal{Y}_{(N,\mathsf{par})}$, EncKey outputs a vector of polynomials $\mathbf{k} = (k_1, \ldots, k_{m_3})$ in non-lone variables and $\mathbf{r} = (r_0, r_1, \ldots, r_{m_1})$ and lone variables α and $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_{m_2})$. For $\ell \in [m_3]$, where $\phi_\ell, \phi_{\ell,z}, \phi_{\ell,i,j} \in \mathbb{Z}_N$, the ℓth polynomial is given by

$$k_\ell(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}) = \phi_\ell \alpha + \sum_{z \in [m_2]} \phi_{\ell,z} \hat{r}_z + \sum_{i \in [m_1]^+} \sum_{j \in [n]} \phi_{\ell,i,j} r_i b_j.$$

Pair$(N, x, y) \to (\mathrm{E}, \hat{\mathrm{E}})$**.** On input N and both x and y, Pair outputs two matrices E and $\hat{\mathrm{E}}$ of size $(w_1 + 1) \times m_3$ and $w_3 \times (m_1 + 1)$, respectively.

For clarity, in cases where the specific MA-PES that is being used is relevant, we index the algorithms by the authority that chooses to use the scheme, *e.g.*, $\mathsf{EncCt}_a(N, x)$ or $\mathsf{EncKey}_a(N, y)$.

Correctness An MA-PES is correct if for every $\kappa = (N, \mathsf{par}), x \in \mathcal{X}_\kappa, y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = \textsc{true}$, the following holds symbolically,

$$\mathbf{s}\mathrm{E}\mathbf{k}^\intercal + \mathbf{c}\hat{\mathrm{E}}\mathbf{r}^\intercal = \alpha s_0 - \omega r_0.$$

Note that in this extended definition EncCt and EncKey are up to the variable names identically defined. Furthermore, if we set $\omega = 0$, then we have the definition of pair encodings back as defined by [AC17] (except for the extra term r_0, however, we can see this as an alternative numbering of the components in $\mathbf{r}$).

5.4.1 Multi-authority Admissible Pair Encoding Security

For a multi-authority pair encoding scheme to be secure, we require *statistical security*, similar to the *perfect security* notion by Attrapadung [Att14]. For the security of the encoding, it is helpful to realize that we will apply the

dual system encryption technique by (partially) replicating the scheme in the various subgroups. The security properties of the encoding will be used in the semi-functional subgroups, allowing us to prove indistinguishability among several variants of semi-functional ciphertexts and keys.

Instead of requiring that the value α is hidden in the adversary's view, as required in a PES, we require, as a security property for our MA-PES, that the value ω is hidden in the adversary's view. This property allows us to prove that an adversary cannot distinguish a correctly distributed challenge ciphertext from a challenge ciphertext taken from a more restricted distribution. The property should hold even if user secret keys are given, but only as long as the values y associated to these keys do not let the predicate evaluate to TRUE.

Definition 14 (Statistical Security)**.** A multi-authority admissible pair encoding scheme (MA-PES) is *statistically secure* for $\kappa = (N, \mathsf{par}) \in \mathbb{N}^c$, if for all $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$, the values $(w_1, w_2, \mathbf{c}(\omega, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b})) \leftarrow \mathsf{EncCt}(N, x)$ and $(m_1, m_2, \mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b})) \leftarrow \mathsf{EncKey}(N, y)$, if $P_\kappa(x, y) = \text{FALSE}$, the distributions

$$\{\mathbf{s}, \mathbf{c}(0, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}), \mathbf{r}, \mathbf{k}(0, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b})\} \quad \text{and} \quad \{\mathbf{s}, \mathbf{c}(\omega, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}), \mathbf{r}, \mathbf{k}(0, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b})\}$$

are statistically indistinguishable, where the probability is taken over $\mathbf{b} \xleftarrow{R} \mathbb{Z}_p^n$, $\omega \xleftarrow{R} \mathbb{Z}_p$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_p^{(w_1+1)}$, $\hat{\mathbf{s}} \xleftarrow{R} \mathbb{Z}_p^{w_2}$, $\mathbf{r} \xleftarrow{R} \mathbb{Z}_p^{(m_1+1)}$, and $\hat{\mathbf{r}} \xleftarrow{R} \mathbb{Z}_p^{m_2}$ (*i.e.*, the distributions need to be statistically close in the size of p), for every prime $p \mid N$.

In our security proof for the conversion algorithm (see Section 5.6), we additionally need to restrict the output of $\mathsf{EncKey}(N, y)$ of an MA-PES. We require that if, for some $\ell \in [m_3]$, the polynomial k_ℓ contains α, also $r_0 b_1$ needs to be present in the polynomial. More specifically, we require that $\phi_\ell = \phi_{\ell,0,1}$. Note that combining this constraint with the correctness property, we also have that $\eta_\ell = \eta_{\ell,0,1}$.

We give several examples of an MA-PES in Section 5.7.

5.5 Conversion from Encoding to Encryption

A collection of statistically secure MA-PESS can be converted to a fully secure MA-PE scheme using a generic algorithm.

The encryption algorithm can be seen as a combination of the encryption algorithms of several (modified) PE schemes. First, we encrypt a message $m \in \mathbb{G}_T$ by blinding the message with a random element $e(g_1, g_1)^\Delta$. Next, we (additively) secret share Δ into shares δ_a for each of the involved authorities $a \in \mathcal{A}$. For each authority, we encrypt the value $e(g_1, g_1)^{\delta_a}$ using

the randomness $\alpha_a s_{a,0}$. From the correctness of the MA-PES, we know that a user having the appropriate keys can combine the ciphertext and keys in such a way that it obtains the value $\alpha_a s_{a,0} - \omega_a r_0$. Hence, the user can recover the value $e(g_1, g_1)^{\delta_a}$ up to a newly introduced random element that has $\omega_a r_0$ in the exponent. We use this randomness $\omega_a r_0$ to prevent user collusion. Recall that EncCt determines the value ω_a, while EncKey determines the value r_0. So, if we additively secret share 0 into the values ω_a and choose a fixed value r_0 for each ID, we have that, only if a user is able to obtain $e(g_1, g_1)^{\delta_a + \omega_a r_0}$ for all *all* authorities a, the user can combine these values to obtain the randomness used in the encryption of the message m, $e(g_1, g_1)^{\sum_a \delta_a + 0} = e(g_1, g_1)^{\Delta}$.

Although our employed technique is similar to conversion algorithms used in single authority predicate encryption (SA-PE) [CW14; AC16; AC17], we use the fact that the symbol ω, an element part of the *ciphertext*, is statistically hidden. In contrast, SA-PE requires α, an element part of a *key*, to be statistically hidden. Therefore, in our employed proof technique, we can only randomize ω as part of the ciphertext and not α as part of the keys. As an consequence, we require a composite-order bilinear group with three subgroups, instead of the common two subgroups. This also implies that we cannot use the existing constructions for dual system groups [CW14; AC16].

In our construction, we require that identities are random elements from the identity space $\mathcal{ID} = \mathbb{G}$. We achieve this by choosing a cryptographic hash function $H\colon \{0,1\}^* \to \mathbb{G}$ and hash the ID to obtain a random element in $\mathbb{G}$. In our security proof, we require that the challenger can decide on the image of $H(\mathsf{ID})$, $\mathrm{Im}(H) = \mathbb{G}' \subseteq \mathbb{G}$. This requirement is fulfilled by proving the construction secure in the programmable random oracle model.

$\mathsf{GlobalSetup}(1^\lambda)$. The $\mathsf{GlobalSetup}$ algorithm first runs $\mathcal{G}_1(1^\lambda)$ to obtain the group parameters $\mathsf{GP} = (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e, g)$ and $g_1 \xleftarrow{R} \mathbb{G}_1$. It sets the message space $\mathcal{M} = \mathbb{G}_T$ and the identity space $\mathcal{ID} = \mathbb{G}$. It defines a hash function $H\colon \{0,1\}^* \to \mathbb{G}$ and outputs (GP, g_1, H) as the global public parameters pp.

$\mathsf{AuthoritySetup}(\mathsf{pp}, \mathsf{par}_a)$. Given par_a for an MA-PES, the algorithm runs $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n. It picks $\mathbf{v} \xleftarrow{R} \mathbb{Z}_N^n$ and $\alpha \xleftarrow{R} \mathbb{G}_1$, and sets $\mathsf{sk} = g_1^\alpha$. The authority's pk is $(g_1^{\mathbf{v}}, e(g_1, \mathsf{sk}))$. The authority's msk is the tuple $(\mathbf{v}, \mathsf{sk})$.

$\mathsf{Encrypt}(\{(\mathsf{pk}_a, x_a)\}_{a \in \mathcal{A}}, m)$. Choose an $a' \in \mathcal{A}$, pick $\omega_a \xleftarrow{R} \mathbb{Z}_N$ for each authority $a \in \mathcal{A} \setminus a'$, and set $\omega_{a'} = -\sum_{a \in \mathcal{A} \setminus a'} \omega_a$. Additionally, pick $\delta_a \xleftarrow{R} \mathbb{Z}_N$ for all $a \in \mathcal{A}$ and define $e(g_1, g_1)^{\Delta} = \prod_{a \in \mathcal{A}} e(g_1, g_1)^{\delta_a}$. Blind the message $m \in \mathbb{G}_T$ using $e(g_1, g_1)^{\Delta}$ to obtain $\mathsf{ct}_0 = m \cdot e(g_1, g_1)^{\Delta}$.

Now, for each authority $a \in \mathcal{A}$ continue as follows (we frequently

drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(N, x)$ to obtain w_1, w_2, and polynomials $(c_1, \ldots, c_{w_3})$. For $k \in [w_1 + w_2]^+$, pick $s_{a,k} \in \mathbb{Z}_N$, and set $\mathsf{ct}_{a,1,i} = g_1^{s_{a,i}}$ for $i \in [w_1]^+$ and

$$\mathsf{ct}_{a,2,\ell} = (g_1^{\omega_a})^{\eta_\ell} \cdot \prod_{z \in [w_2]} g_1^{\eta_{\ell,z} s_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} \left(g_1^{v_j}\right)^{\eta_{\ell,i,j} s_{a,i}}$$

for $\ell \in [w_3]$. Blind the value $e(g_1, g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1, g_1)^{\delta_a} \cdot e(g_1, \mathsf{sk}_a)^{s_{a,0}}$.

The complete ciphertext is

$$\mathsf{ct} = \left(\mathsf{ct}_0, \left\{\mathsf{ct}_{a,0}, \mathsf{ct}_{a,1,0}, \ldots, \mathsf{ct}_{a,1,w_1}, \mathsf{ct}_{a,2,1}, \ldots, \mathsf{ct}_{a,2,w_3}\right\}_{a \in \mathcal{A}}\right).$$

KeyGen$(\mathsf{msk}_a, y, \mathsf{ID})$**.** The algorithm $\mathsf{EncKey}_a(N, y)$ is run to obtain m_1, m_2, and polynomials $(k_1, \ldots, k_{m_3})$. Set $\mathsf{usk}_{a,1,0} = H(\mathsf{ID})$ and pick $r_i \xleftarrow{R} \mathbb{Z}_N$ to set $\mathsf{usk}_{a,1,i} = g_1^{r_i}$ for $i \in [m_1 + m_2]$. Set

$$\mathsf{usk}_{a,2,\ell} = \mathsf{sk}_a^{\phi_\ell} \cdot \prod_{z \in [m_2]} \left(\mathsf{usk}_{a,1,m_1+z}\right)^{\phi_{\ell,z}} \cdot \prod_{i \in [m_1]^+, j \in [n]} \left(\mathsf{usk}_{a,1,i}^{v_j}\right)^{\phi_{\ell,i,j}}$$

for $\ell \in [m_3]$. The complete user secret key for $y \in \mathcal{Y}_{\kappa(a)}$ is

$$\mathsf{usk}_{y,\mathsf{ID}} = \left(\mathsf{usk}_{a,1,0}, \ldots, \mathsf{usk}_{a,1,m_1}, \mathsf{usk}_{a,2,1}, \ldots, \mathsf{usk}_{a,2,m_3}\right).$$

Note that $\mathsf{usk}_{a,1,m_1+z}$ for $z \in [m_2]$ are *not* included in the complete usk.

Decrypt$(\{\mathsf{usk}_{y,\mathsf{ID}}\}_y, \mathsf{ct})$**.** To decrypt the ciphertext ct, we first decrypt $\mathsf{ct}_{a,0}$ for each authority $a \in \mathcal{A}$. Run $\mathsf{Pair}_a(N, x_a, y_a)$ to obtain E_a and $\hat{\mathrm{E}}_a$. Now compute

$$\begin{aligned}
&\mathsf{ct}_{a,0} \cdot \left(\prod_{\substack{i \in [w_1]^+, \\ \ell \in [m_3]}} e(\mathsf{ct}_{a,1,i}, \mathsf{usk}_{a,2,\ell})^{\mathrm{E}_{a,i,\ell}} \cdot \prod_{\substack{\ell \in [w_3], \\ i \in [m_1]^+}} e(\mathsf{ct}_{a,2,\ell}, \mathsf{usk}_{a,1,i})^{\hat{\mathrm{E}}_{a,\ell,\mathrm{i}}} \right)^{-1} \\
&= \left(e(g_1, g_1)^{\delta_a} \cdot e(g_1, \mathsf{sk}_a)^{s_{a,0}}\right) \left(e(g_1, g_1)^{\alpha_a s_{a,0} - \omega_a r_0}\right)^{-1} \\
&= e(g_1, g_1)^{\delta_a} \cdot e(g_1, g_1)^{\alpha s_{a,0}} \cdot e(g_1, g_1)^{-\alpha_a s_{a,0} + \omega_a r_0} \\
&= e(g_1, g_1)^{\delta_a} \cdot e(g_1, g_1)^{\omega_a r_0}
\end{aligned}$$

for some value r_0 independent of a. We can now combine these results to obtain

$$\begin{aligned}
\prod_{a \in \mathcal{A}} \left(e(g_1, g_1)^{\delta_a} \cdot e(g_1, g_1)^{\omega_a r_0}\right) &= e(g_1, g_1)^{\sum_{a \in \mathcal{A}} \delta_a} \cdot e(g_1, g_1)^{\sum_{a \in \mathcal{A}} \omega_a r_0} \\
&= e(g_1, g_1)^{\Delta} \cdot e(g_1, g_1)^{0 r_0} \\
&= e(g_1, g_1)^{\Delta},
\end{aligned}$$

and recover the plaintext $m = \mathsf{ct}_0 \cdot e(g_1, g_1)^{-\Delta}$.

Remark 1 (One-Use Requirement)**.** If the values $\mathbf{b}$ of an MA-PES are used multiple times in the same ciphertext, they might not be statistically hidden anymore and information on ω might be leaked. Therefore, if we want to make sure to avoid using (part) of the same $\mathbf{b}$ multiple times, we may require that an authority may occur *only once* in a ciphertext of a corresponding MA-PE scheme. Such a requirement is similar to the *one-use requirement* as found in several ABE schemes [LOS+10; LW11; Att14] where the *attributes* may only occur once.

Remark 2 (Type of Secret Sharing)**.** Instead of using additive secret sharing as described above, we could have also decided to use SSS. By using SSS, we allow for combining the predicates from different authorities in the ciphertext using both AND and OR gates—like in the MA-ABE scheme by Lewko and Waters [LW11]—while additive secret sharing only allows for combining them using AND gates. However, we can easily emulate OR gates by writing the desired combination of predicates for different authorities in DNF and creating a new ciphertext for each of the conjunctive clauses. The main advantage of choosing to use additive secret sharing, is that it simplifies the construction and the corresponding security proofs.

5.6 Security of the Conversion Algorithm

We prove security similarly to the dual system encryption technique [Wat09] variant that was used to prove MA-ABE secure before [LW11]. As such, we first introduce semi-functional ciphertext and semi-functional keys. These semi-functional ciphertexts and keys are solely used in the security proofs and not in the actual scheme.

5.6.1 Semi-functional Ciphertext

A semi-functional ciphertext can be created by slightly modifying the encryption algorithm for normal ciphertexts as given before. We define the various types of semi-functional ciphertext through the algorithm $\overline{\mathsf{Encrypt}}$.

$\overline{\mathsf{Encrypt}}(\{(\mathsf{pk}_a, x_a)\}_{a\in\mathcal{A}}, m; \mathcal{C}, \{\mathsf{sk}_a\}_{a\in\mathcal{A}})$**.** This algorithm is similar to Encrypt, but also takes a set $\mathcal{C} \subseteq \{1,2,3\}$ and the authorities' sk as input.

While in normal ciphertext, we use $g_1^{\omega_a}$, where $\sum_{a\in\mathcal{A}} \omega_a = 0$, in semi-functional ciphertext, we use $g_1^{\omega_{a,1}} g_2^{\omega_{a,2}} g_3^{\omega_{a,3}}$ and require $\sum_{a\in\mathcal{A}} \omega_{a,i} = 0$ only for $i \in \mathcal{C}$. For the values $i \in \{1,2,3\} \setminus \mathcal{C}$, we pick $\omega_{a,i} \xleftarrow{R} \mathbb{Z}_N$ without any constraint on the sum of these values.

Additionally, the construction of the values $\mathsf{ct}_{a,1,i}$ and $\mathsf{ct}_{a,2,\ell}$ is dependent on whether the authority a was created by the challenger (*i.e.*, $a \in$

I) or by the adversary (*i.e.*, $a \in \bar{I}$). If $a \in I$, all of the encoding variables $(\mathbf{s}_a, \mathbf{c}_a(\omega_a, \mathbf{s}_a, \hat{\mathbf{s}}_a, \mathbf{b}_a)$ are mapped to elements in $\mathbb{G}$. However, if $a \in \bar{I}$, only ω is mapped to an element in $\mathbb{G}$ (*i.e.*, $g_1^{\omega_{a,1}} g_2^{\omega_{a,2}} g_3^{\omega_{a,3}}$), while all other encoding variables are mapped to elements in $\mathbb{G}_1 \subset \mathbb{G}$ just like in normal ciphertext.

In the proofs, we will use several types of semi-functional ciphertext. We use $\overline{\mathsf{Encrypt}}$ for $\mathcal{C} = \{1,2,3\}$, $\mathcal{C} = \{1,2\}$, and $\mathcal{C} = \{1\}$.

Pseudo Normal Ciphertext In case we use $\mathcal{C} = \{1,2,3\}$, we say that the ciphertext is *pseudo normal*.

Nominally Semi-function Ciphertext In case we use $\mathcal{C} = \{1,2\}$, we say that the ciphertext is *nominally semi-functional*.

5.6.2 Semi-functional Keys

Besides normal keys, we define *pseudo normal* keys and two types of semi-functional keys. We can conveniently define these non-normal keys through the algorithm $\overline{\mathsf{KeyGen}}$.

$\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; g', r_0)$. The algorithm is similarly defined as the original $\mathsf{KeyGen}(\mathsf{msk}_a, y, \mathsf{ID})$, however, instead of using the generator g_1 and the hash function $H\colon \{0,1\}^* \to \mathbb{G}$, the generator g' and the function $H\colon \mathsf{ID} \mapsto (g')^{r_0}$ are used. As a consequence, all elements of $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; g', r_0)$ are elements of the group $\langle g' \rangle$.

Normal Key Note that a normal key cannot be described using $\overline{\mathsf{KeyGen}}$: While we can set $g' \in \mathbb{G}_1$, the hash function H is defined as $H\colon \{0,1\}^* \to \mathbb{G}$ and not as $H\colon \{0,1\}^* \to \mathbb{G}_1$.

Pseudo Normal Key A pseudo normal key is created using $\overline{\mathsf{KeyGen}}$ with $g' \in \mathbb{G}_1$. It differs from a normal key in that H maps to an element in $\mathbb{G}_1$, $H\colon \{0,1\} \to \mathbb{G}_1$, instead of mapping to an element in $\mathbb{G}$.

Semi-functional Key of Type I A semi-functional key of type I is created using $\overline{\mathsf{KeyGen}}$ with $g' = g_1 g_2$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

Semi-functional Key of Type II A semi-functional key of type II is created using $\overline{\mathsf{KeyGen}}$ with $g' = g_1 g_3$, where $g_1 \in \mathbb{G}_1$ and $g_3 \in \mathbb{G}_3$.

Game	Challenge ciphertext $\mathsf{ct}_{\mathbf{x}^*}$	Queried key $\mathsf{usk}_{\mathbf{y},\mathsf{ID}}$
original	$\mathsf{Encrypt}(\{\mathbf{x}^*\}, m_b)$	$\mathsf{KeyGen}(\mathbf{y})$
0	$\mathsf{Encrypt}(\{\mathbf{x}^*\}, m_b)$	$\overline{\mathsf{KeyGen}}(\mathbf{y}; \boxed{g_1}, u_{\mathsf{ID}})$
1	$\boxed{\overline{\mathsf{Encrypt}}(\{\mathbf{x}^*\}, m_b; \{1,2,3\}, \{\mathsf{sk}\})}$	$\overline{\mathsf{KeyGen}}(\mathbf{y}; g_1, u_{\mathsf{ID}})$
2,j,1	$\overline{\mathsf{Encrypt}}(\{\mathbf{x}^*\}, m_b; \boxed{\{1,2\}}, \{\mathsf{sk}\})$	$\overline{\mathsf{KeyGen}}(\mathbf{y}; \boxed{g_{12}}, u_{\mathsf{ID}})$
2,j,2	$\overline{\mathsf{Encrypt}}(\{\mathbf{x}^*\}, m_b; \boxed{\{1\}}, \{\mathsf{sk}\})$	$\overline{\mathsf{KeyGen}}(\mathbf{y}; \boxed{g_{13}}, u_{\mathsf{ID}})$
3	$\overline{\mathsf{Encrypt}}(\{\mathbf{x}^*\}, \boxed{\text{random}}; \{1\}, \{\mathsf{sk}\})$	$\overline{\mathsf{KeyGen}}(\mathbf{y}; g_{13}, u_{\mathsf{ID}})$

Figure 5.2. Summary of the sequence of games used in the proof. An explanation of the difference between the games is given in Section 5.6.3.

5.6.3 Hybrids and Proof Outline

We will prove security through a series of hybrid games. Let $\mathsf{Game}_{\mathsf{original}}$ be the original full security game as defined in Definition 12. Game_0 is defined similarly, except that in this game only pseudo normal keys are used, by both the challenger and the adversary, instead of normal keys. In Game_1 the challenger answers the challenge query with a semi-functional ciphertext instead of a normal ciphertext as used in Game_0. Let q denote the number of distinct IDs for which the adversary queries keys for. We define two types of games for each j from 1 to q. In $\mathsf{Game}_{2,j,1}$, the queries for the first $j-1$ identities are answered with semi-functional keys of type II, while key queries for the jth identity are answered with a semi-functional key of type I. In $\mathsf{Game}_{2,j,2}$, the challenger answers key queries for the first j identities with a semi-functional key of type II. We define Game_3 as the game where all key queries are answered by semi-functional keys of type II and where the challenge ciphertext is replaced by an encryption of a random message.

A summary of the sequence of games can be found in Figure 5.2. In this figure, we also indicate the exact type of semi-functional challenge ciphertext the adversary receives by specifying the input $\mathcal{C}$ to $\overline{\mathsf{Encrypt}}$. In the cases where the values $\omega_{a,2}$ or $\omega_{a,3}$ sum to a random value (*i.e.*, $\mathcal{C} = \{1,2\}$ and $\mathcal{C} = \{1\}$), we have to show that the adversary cannot distinguish this from the case where the values $\omega_{a,2}$ and $\omega_{a,3}$ are guaranteed to sum to zero (*i.e.*, $\mathcal{C} = \{1,2,3\}$).

For example, in the hybrid from $\mathsf{Game}_{2,j,1}$ to $\mathsf{Game}_{2,j,2}$, we have to show that the adversary cannot distinguish a ciphertext created with $\sum_{a\in\mathcal{A}^*} \omega_{a,2} = 0$ from a ciphertext created with $\sum_{a\in\mathcal{A}^*} \omega_{a,2} \in_R \mathbb{Z}_{p_2}$. In this case, we know that $P(\{x^*_a\}_{a\in\mathcal{A}^*}, \{y_{\mathsf{ID},a}\}_{a\in\mathcal{A}^*}) = \textsc{false}$, *i.e.*, there exists at least one $a' \in \mathcal{A}^*$ such that $P_{\kappa(a')}(x^*_{a'}, y_{\mathsf{ID},a'}) = \textsc{false}$ or no query for $(a', y_{a'}, \mathsf{ID})$ has been

made. Furthermore, observe that the value $\omega_{a',2}$ only occurs in the ciphertext part $(\mathsf{ct}_{a',2,0}, \ldots, \mathsf{ct}_{a',2,w_3})$ of authority a', corresponding to the values $\mathbf{c}_{a'}$ of $\mathsf{EncCt}_{a'}$. By the statistical security requirement (see Definition 14), we know that this $\omega_{a',2}$ is statistically hidden in the adversary's view. From this fact, it clearly follows that the sum of all $\omega_{a,2}$ (*i.e.*, $\sum_{a \in \mathcal{A}^*} \omega_{a,2}$) includes $\omega_{a',2}$ and thus the value of the sum is statistically hidden in the adversary's view as well. Hence, the adversary cannot distinguishing whether it received a ciphertext where the $\omega_{a,2}$ are shares of zero, or independently random shares.

In $\mathsf{Game}_{2,q,2}$, all key queries are answered with a type II key, and we know that the values $\omega_{a,3}$ do not need to sum to 0. Since there are no further constraints on $\omega_{a,3}$, we can set all $\omega_{a,3} \xleftarrow{R} \mathbb{Z}_N$. Thus, we essentially have that an adversary cannot distinguish whether the ciphertext components for *any* authority have been randomized or not. We use this fact to show that the sum of the values δ_i, as appearing in the semi-functional ciphertext, is computationally indistinguishable from random as well.

We prove indistinguishability of the hybrids using several lemmas. Combining Lemmas 2 to 6 proves the following theorem.

Theorem 9. *For any collection of predicate families for authorities $a \in \mathcal{A}$, $P_a = \{P_{\kappa(a)}\}_{\kappa(a) \in \mathbb{N}^c}$, if each* MA-PES *for $P_{\kappa(a)}$ satisfies $\phi_\ell = \phi_{\ell,0,1}$ for all $\ell \in [m_3]$ and is statistically secure (see Definition 14), then the* MA-PE *scheme converted from these* MA-PESS *(see Section 5.5) is fully secure (see Definition 12) in the random oracle model under Assumptions 4 to 7.*

Lemma 2 ($\mathsf{Game}_{\mathsf{original}} \approx_c \mathsf{Game}_0$)**.** *Any adversary $\mathcal{A}$ having at most a negligible advantage in breaking Assumption 4, has at most a negligible advantage in distinguishing $\mathsf{Game}_{\mathsf{original}}$ from Game_0.*

Proof. The challenger $\mathcal{B}$ receives $\{(\mathsf{GP}, g_1), T\}$ as input, where either $T \in_R \mathbb{G}$ or $T \in_R \mathbb{G}_1$. Now, $\mathcal{B}$ plays the following game with $\mathcal{A}$.

Hash Oracle Upon receiving oracle query ID for the hash function H, the challenger $\mathcal{B}$ checks if it received the query before, and if so, answers with the same reply as before. If $\mathcal{A}$ has not queried for the hash value of ID before, $\mathcal{B}$ picks a value $u_{\mathsf{ID}} \xleftarrow{R} \mathbb{Z}_N$ and replies with $T^{u_{\mathsf{ID}}}$.

Setup The challenger $\mathcal{B}$ sets $\mathsf{pp} = (\mathsf{GP}, g_1)$ and sends pp to the adversary $\mathcal{A}$.

Authority Queries Request for a new authority a using par_a are answered by the challenger by running $\mathsf{AuthoritySetup}(\mathsf{pp}, \mathsf{par}_a)$. The challenger first uses $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n, picks $\mathbf{v} \xleftarrow{R} \mathbb{Z}_N^n$ and $\alpha \xleftarrow{R} \mathbb{G}_1$, and

sets $\mathsf{sk} = g_1^{\alpha}$. It sets the public key pk as $(g_1^{\mathbf{v}}, e(g_1, \mathsf{sk}))$ and the authority secret key msk as $(\mathbf{v}, \mathsf{sk})$. It sends pk to the adversary and adds a to the set I.

Key Queries Upon receiving a key query $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$ for an uncorrupted authority $a \in I$, $\mathcal{B}$ answers the query by first running $\mathsf{EncKey}_a(N, y)$ to obtain m_1, m_2, and polynomials $(k_1, \ldots, k_{m_3})$. Next, it sets $\mathsf{usk}_{a,1,0} = T^{u_{\mathsf{ID}}}$ and picks $r_i \xleftarrow{R} \mathbb{Z}_N$ for $i \in [m_1 + m_2]$ to set $\mathsf{usk}_{a,1,i} = g_1^{r_i}$ for $i \in [m_1]$. Additionally, it sets

$$\mathsf{usk}_{a,2,\ell} = \mathsf{sk}_a^{\phi_\ell} \cdot \prod_{z \in [m_2]} \left(\mathsf{usk}_{a,1,m_1+z}\right)^{\phi_{\ell,z}} \cdot \prod_{i \in [m_1]^+, j \in [n]} \left(\mathsf{usk}_{a,1,i}^{v_j}\right)^{\phi_{\ell,i,j}}$$

for $\ell \in [m_3]$. Finally, it returns the secret key for $y \in \mathcal{Y}_{\kappa(a)}$ as

$$\mathsf{usk}_{y,\mathsf{ID}} = (\mathsf{usk}_{a,1,0}, \ldots, \mathsf{usk}_{a,1,m_1}, \mathsf{usk}_{a,2,1}, \ldots, \mathsf{usk}_{a,2,m_3}).$$

Challenge Ciphertext Whenever $\mathcal{A}$ requests the ciphertext challenge by sending $(m_0, m_1, \{x_a^*\}_{a \in \mathcal{A}^*})$ along with the public keys $\{\mathsf{pk}_a\}_{a \in \mathcal{A}^* \cap \bar{I}}$, the challenger $\mathcal{B}$ picks $b \xleftarrow{R} \{0, 1\}$ and encrypts message m_b as a normal challenge ciphertext using

$$\mathsf{Encrypt}(\{\mathsf{pk}_a\}_{a \in \mathcal{A}^*}, \{x_a^*\}_{a \in \mathcal{A}^*}, m_b).$$

Now, observe that $\mathcal{A}$ is playing $\mathsf{Game}_{\mathsf{original}}$ if $T \in_R \mathbb{G}$, while it is playing Game_0 if $T \in_R \mathbb{G}_1$. Therefore, if $\mathcal{A}$ has a non-negligible advantage in deciding which game it is playing, $\mathcal{B}$ has a non-negligible advantage in breaking Assumption 4. □

Lemma 3 ($\mathsf{Game}_0 \approx_c \mathsf{Game}_1$)**.** *Any adversary $\mathcal{A}$ having at most a negligible advantage in breaking Assumption 4, has at most a negligible advantage in distinguishing Game_0 from Game_1.*

Proof. The challenger $\mathcal{B}$ receives $\{(\mathrm{GP}, g_1), T\}$ as input, where either $T \in_R \mathbb{G}$ or $T \in_R \mathbb{G}_1$. Now, $\mathcal{B}$ plays the game with $\mathcal{A}$ as follows.

Hash Oracle Upon receiving oracle query ID for the hash function H, the challenger $\mathcal{B}$ checks if it received the query before, and if so, answers with the same reply as before. If $\mathcal{A}$ has not queried for the hash value of ID before, $\mathcal{B}$ picks a value $u_{\mathsf{ID}} \xleftarrow{R} \mathbb{Z}_N$ and replies with $g_1^{u_{\mathsf{ID}}}$.

Setup The challenger $\mathcal{B}$ sets $\mathsf{pp} = (\mathrm{GP}, g_1)$ and sends pp to the adversary $\mathcal{A}$.

Authority Queries Request for a new authority a using par_a are answered by the challenger by running $\mathsf{AuthoritySetup}(\mathsf{pp}, \mathsf{par}_a)$. The challenger first uses $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n, picks $\mathbf{v} \xleftarrow{R} \mathbb{Z}_N^n$ and $\alpha \xleftarrow{R} \mathbb{G}_1$, and sets $\mathsf{sk} = g_1^{\alpha}$. It sets the public key pk as $(g_1^{\mathbf{v}}, e(g_1, \mathsf{sk}))$ and the authority secret key msk as $(\mathbf{v}, \mathsf{sk})$. It sends pk to the adversary and adds a to the set I.

Key Queries Upon receiving a key query $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$ for an uncorrupted authority $a \in I$, $\mathcal{B}$ answers the query using a pseudo normal key using u_{ID} as r_0, $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; g_1, u_{\mathsf{ID}})$.

Challenge Ciphertext Whenever $\mathcal{A}$ requests the ciphertext challenge by sending $(m_0, m_1, \{x_a^*\}_{a \in \mathcal{A}^*})$, the challenger $\mathcal{B}$ picks $b \xleftarrow{R} \{0,1\}$ and encrypts message m_b as a challenge ciphertext using T.

Choose an $a' \in \mathcal{A}^*$, pick $\omega_a \xleftarrow{R} \mathbb{Z}_N$ for each authority $a \in \mathcal{A}^* \setminus a'$, and set $\omega_{a'} = -\sum_{a \in \mathcal{A}^* \setminus a'} \omega_a$. Additionally, pick $\delta_a \xleftarrow{R} \mathbb{Z}_N$, set $e(g_1, g_1)^{\delta_a}$ for all $a \in \mathcal{A}^*$, and define $e(g_1, g_1)^{\Delta} = \prod_{a \in \mathcal{A}^*} e(g_1, g_1)^{\delta_a}$. Blind the message $m_b \in \mathbb{G}_T$ using $e(g_1, g_1)^{\Delta}$ to obtain $\mathsf{ct}_0 = m_b \cdot e(g_1, g_1)^{\Delta}$.

Now, for each authority $a \in \mathcal{A}^*$ continue as follows (we frequently drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(N, x)$ to obtain w_1, w_2, and polynomials $(c_1, \ldots, c_{w_3})$.

If $a \in I$, pick $\tilde{s}_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1 + w_2]^+$, and set $\mathsf{ct}_{a,1,i} = T^{\tilde{s}_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = (T^{\omega_a})^{\eta_\ell} \cdot \prod_{z \in [w_2]} T^{\eta_{\ell,z} \tilde{s}_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} T^{\eta_{\ell,i,j} \tilde{s}_{a,i} v_j}.$$

Blind the value $e(g_1, g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1, g_1)^{\delta_a} \cdot e(T^{\tilde{s}_{a,0}}, g_1^{\alpha_a})$.

If $a \in \bar{I}$, pick $s_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1 + w_2]^+$, and set $\mathsf{ct}_{a,1,i} = g_1^{s_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = (T^{\omega_a})^{\eta_\ell} \cdot \prod_{z \in [w_2]} g_1^{\eta_{\ell,z} s_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} \left(g_1^{v_j}\right)^{\eta_{\ell,i,j} s_{a,i}}.$$

Blind the value $e(g_1, g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1, g_1)^{\delta_a} \cdot e(g_1, \mathsf{sk}_a)^{s_{a,0}}$.

The complete challenge ciphertext is

$$\mathsf{ct} = \left(\mathsf{ct}_0, \{\mathsf{ct}_{a,0}, \mathsf{ct}_{a,1,0}, \ldots, \mathsf{ct}_{a,1,w_1}, \mathsf{ct}_{a,2,1}, \ldots, \mathsf{ct}_{a,2,w_3}\}_{a \in \mathcal{A}^*}\right).$$

Note that $T = g_1^{t \pmod{p_1}} g_2^{t \pmod{p_2}} g_3^{t \pmod{p_3}}$ for unknown t, and so we have implicitly used $s_{a,i} = t\tilde{s}_{a,i}$ in $\mathsf{ct}_{a,2,i}$, making the ciphertext identically distributed to a normal ciphertext if $T \in \mathbb{G}_1$. Moreover, we have $\omega'_{a,1} = t\omega_a$

$(\text{mod } p_1)$, $\omega'_{a,2} = t\omega_a \pmod{p_2}$, and $\omega'_{a,3} = t\omega_a \pmod{p_3}$. Thus, if $T \in_R \mathbb{G}_1$ the resulting ciphertext is normal, while if $T \in_R \mathbb{G}$, the resulting ciphertext is pseudo normal, with $\sum_{a\in\mathcal{A}^*} \omega'_{a,1} = \sum_{a\in\mathcal{A}^*} \omega'_{a,2} = \sum_{a\in\mathcal{A}^*} \omega'_{a,3} = 0$. Moreover, depending on the value of T, $\mathcal{B}$ either plays Game_0 or Game_1. □

Observe that, by definition, $\mathsf{Game}_1 \equiv \mathsf{Game}_{2,0,2}$.

Lemma 4 ($\mathsf{Game}_{2,j-1,2} \approx_c \mathsf{Game}_{2,j,1}$)**.** *Any adversary $\mathcal{A}$ having at most a negligible advantage in breaking Assumption 5, has at most a negligible advantage in distinguishing $\mathsf{Game}_{2,j-1,2}$ from $\mathsf{Game}_{2,j,1}$.*

Proof. The challenger $\mathcal{B}$ receives $\{(\text{GP}, g_1, h_1h_2, g_3), T\}$ as input, where either $T \in_R \mathbb{G}_1$ or $T \in_R \mathbb{G}_{12}$. Now, $\mathcal{B}$ plays the game with $\mathcal{A}$ as follows.

Hash Oracle Upon receiving oracle query ID for the hash function H, the challenger $\mathcal{B}$ checks if it received the query before, and if so, answers with the same reply as before. If $\mathcal{A}$ has not queried for the hash value of ID before, $\mathcal{B}$ picks a value $u_{\mathsf{ID}} \xleftarrow{R} \mathbb{Z}_N$. Then, the first $j-1$ queries for some ID are answered with $(g_1g_3)^{u_{\mathsf{ID}}}$, the jth query is answered with $T^{u_{\mathsf{ID}}}$, while other queries are answered with $g_1^{u_{\mathsf{ID}}}$.

Setup The challenger $\mathcal{B}$ sets $\mathsf{pp} = (\text{GP}, g_1)$ and sends pp to the adversary $\mathcal{A}$.

Authority Queries Request for a new authority a using par_a are answered by the challenger by running $\mathsf{AuthoritySetup}(\mathsf{pp}, \mathsf{par}_a)$. The challenger first uses $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n, picks $\mathbf{v} \xleftarrow{R} \mathbb{Z}_N^n$ and $\alpha \xleftarrow{R} \mathbb{G}_1$, and sets $\mathsf{sk} = g_1^\alpha$. It sets the public key pk as $(g_1^{\mathbf{v}}, e(g_1, \mathsf{sk}))$ and the authority secret key msk as $(\mathbf{v}, \mathsf{sk})$. It sends pk to the adversary and adds a to the set I.

Key Queries Upon receiving a key query $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$ for an uncorrupted authority $a \in I$, $\mathcal{B}$ answers the query depending on the number distinct ID that have been queried before. If ID is one of the $(j-1)$th first IDs being queried, $\mathcal{B}$ answers with a semi-functional key of type II by sending $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; g_1g_3, u_{\mathsf{ID}})$. If the query is for the jth ID, $\mathcal{B}$ answers by sending $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; T, u_{\mathsf{ID}})$. Otherwise, $\mathcal{B}$ answers with a pseudo normal key by sending $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; g_1, u_{\mathsf{ID}})$.

Note that all in cases the key queries are answered with elements from the hash oracle's range, creating properly distributed (semi-functional) keys. Also, observe that if $T \in_R \mathbb{G}_1$, a query for the jth ID is answered with a pseudo normal key. Otherwise, if $T \in_R \mathbb{G}_{12}$, the query is answered with a semi-functional key of type I.

Challenge Ciphertext Whenever $\mathcal{A}$ requests the ciphertext challenge by sending $(m_0, m_1, \{x^*_a\}_{a\in\mathcal{A}^*})$, the challenger $\mathcal{B}$ picks $b \xleftarrow{R} \{0,1\}$ and encrypts message m_b as a challenge ciphertext using h_1h_2 and g_3.

Choose an $a' \in \mathcal{A}^*$, pick $\omega'_{a,12} \xleftarrow{R} \mathbb{Z}_N$ for each authority $a \in \mathcal{A}^* \setminus a'$, and set $\omega'_{a',12} = -\sum_{a\in\mathcal{A}^*\setminus a'} \omega'_{a,12}$. Additionally, pick $\omega'_{a,3}, \delta_a \xleftarrow{R} \mathbb{Z}_N$, and set $e(g_1,g_1)^{\delta_a}$ for all $a \in \mathcal{A}^*$, and define $e(g_1,g_1)^{\Delta} = \prod_{a\in\mathcal{A}^*} e(g_1,g_1)^{\delta_a}$. Blind the message $m_b \in \mathbb{G}_T$ using $e(g_1,g_1)^{\Delta}$ to obtain $\mathsf{ct}_0 = m_b \cdot e(g_1,g_1)^{\Delta}$.

Now, for each authority $a \in \mathcal{A}^*$ continue as follows (we frequently drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(N,x)$ to obtain w_1, w_2, and polynomials $(c_1,\ldots,c_{w_3})$.

If $a \in I$, pick $\tilde{s}_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1+w_2]^+$, and set $\mathsf{ct}_{a,1,i} = (h_1h_2g_3)^{\tilde{s}_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = \Big((h_1h_2)^{\omega'_{a,12}}(g_3)^{\omega'_{a,3}}\Big)^{\eta_\ell} \cdot \prod_{z\in[w_2]} (h_1h_2g_3)^{\eta_{\ell,z}\tilde{s}_{a,w_1+z}} \cdot \prod_{i\in[w_1]^+,j\in[n]} (h_1h_2g_3)^{\eta_{\ell,i,j}\tilde{s}_{a,i}v_j}.$$

Blind the value $e(g_1,g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1,g_1)^{\delta_a} \cdot e\big((h_1h_2)^{\tilde{s}_{a,0}}, g_1^{\alpha_a}\big)$.

If $a \in \bar{I}$, pick $s_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1+w_2]^+$, and set $\mathsf{ct}_{a,1,i} = g_1^{s_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = \Big((h_1h_2)^{\omega'_{a,12}}(g_3)^{\omega'_{a,3}}\Big)^{\eta_\ell} \cdot \prod_{z\in[w_2]} g_1^{\eta_{\ell,z}s_{a,w_1+z}} \cdot \prod_{i\in[w_1]^+,j\in[n]} \big(g_1^{v_j}\big)^{\eta_{\ell,i,j}s_{a,i}}.$$

Blind the value $e(g_1,g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1,g_1)^{\delta_a} \cdot e(g_1,\mathsf{sk}_a)^{s_{a,0}}$.

The complete challenge ciphertext is

$$\mathsf{ct} = \big(\mathsf{ct}_0, \{\mathsf{ct}_{a,0}, \mathsf{ct}_{a,1,0}, \ldots, \mathsf{ct}_{a,1,w_1}, \mathsf{ct}_{a,2,1}, \ldots, \mathsf{ct}_{a,2,w_3}\}_{a\in\mathcal{A}^*}\big).$$

To see that this is properly distributed as a nominally semi-functional ciphertext, observe that $\omega'_{a,12} \mod p_1$ is independent of $\omega'_{a,12} \mod p_2$. Moreover, note that (for all i) the values $s_{a,i} \mod p_1$, $s_{a,i} \mod p_2$, and $s_{a,i} \mod p_3$ are mutually independent. So, the given ciphertext is distributed as a nominally semi-functional one, and thus, we are left to prove that adversary $\mathcal{A}$ cannot distinguish a pseudo normal ciphertext (with $\mathcal{C} = \{1,2,3\}$) from a nominally semi-functional ciphertext (with $\mathcal{C} = \{1,2\}$).

Let $a' \in \mathcal{A}^* \cap I$ be an authority for which $\mathcal{A}$ cannot decrypt the ciphertext component $\mathsf{ct}_{a',0}$ because $P_{a'}(x^*_{a'}, y_{a'}) = \textsc{false}$. Such an authority exists as otherwise $\mathcal{A}$ would be able to trivially decrypt the challenge ciphertext. Now,

observe that all values $\omega'_{a,3}$ look random for $a \in \mathcal{A}^* \setminus a'$, while $\omega'_{a',3} \in_R \mathbb{Z}_N$ for nominally semi-functional ciphertext and $\omega'_{a',3} = -\sum_{a \in \mathcal{A}^* \setminus a'} \omega'_{a,3}$ for pseudo normal ciphertext. Hence, $\mathcal{A}$'s view can *at most* contain information about $\omega'_{a,3}$ on the values $\{\mathbf{s}_{a'}, \mathbf{c}_{a'}(0, \mathbf{s}_{a'}, \hat{\mathbf{s}}_{a'}, \mathbf{b}_{a'}), \mathbf{r}_{a'}, \mathbf{k}_{a'}(0, \mathbf{r}_{a'}, \hat{\mathbf{r}}_{a'}, \mathbf{b}_{a'})\}$ in the subgroup $\mathbb{G}_3$ (remember, $P_{a'}(x^*_{a'}, y_{a'}) = \text{FALSE}$ for the $y_{a'}$ of the jth ID). No other information about the values in these subgroups is given by any of the key query responses (note $\mathbf{b}_{a'}$ is independent of $\mathbf{b}_a$). By the statistical security property (see Definition 14), we know that this view is now indistinguishable from $\{\mathbf{s}_{a'}, \mathbf{c}_{a'}(\omega_{a'}, \mathbf{s}_{a'}, \hat{\mathbf{s}}_{a'}, \mathbf{b}_{a'}), \mathbf{r}_{a'}, \mathbf{k}_{a'}(0, \mathbf{r}_{a'}, \hat{\mathbf{r}}_{a'}, \mathbf{b}_{a'})\}$, the view of a nominally semi-functional ciphertext. Hence, the ciphertext is distributed correctly according to the adversary's view. Moreover, depending on the value of T, $\mathcal{B}$ either plays $\mathsf{Game}_{2,j-1,2}$ or $\mathsf{Game}_{2,j,1}$. □

Lemma 5 ($\mathsf{Game}_{2,j,1} \approx_c \mathsf{Game}_{2,j,2}$)**.** *Any adversary $\mathcal{A}$ having at most a negligible advantage in breaking Assumption 6, has at most a negligible advantage in distinguishing $\mathsf{Game}_{2,j,1}$ from $\mathsf{Game}_{2,j,2}$.*

Proof. The challenger $\mathcal{B}$ receives $\{(\text{GP}, g_1, h_1 h_3, h'_2 h'_3), T\}$ as input, where either $T \in_R \mathbb{G}_{12}$ or $T \in_R \mathbb{G}_{13}$. Now, $\mathcal{B}$ plays the game with $\mathcal{A}$ as follows.

Hash Oracle Upon receiving oracle query ID for the hash function H, the challenger $\mathcal{B}$ checks if it received the query before, and if so, answers with the same reply as before. If $\mathcal{A}$ has not queried for the hash value of ID before, $\mathcal{B}$ picks a value $u_{\mathsf{ID}} \xleftarrow{R} \mathbb{Z}_N$. Then, the first $j-1$ queries for some ID are answered with $(h_1 h_3)^{u_{\mathsf{ID}}}$, the jth query is answered with $T^{u_{\mathsf{ID}}}$, while other queries are answered with $g_1^{u_{\mathsf{ID}}}$.

Setup The challenger $\mathcal{B}$ sets $\mathsf{pp} = (\text{GP}, g_1)$ and sends pp to the adversary $\mathcal{A}$.

Authority Queries Request for a new authority a using par_a are answered by the challenger by running $\mathsf{AuthoritySetup}(\mathsf{pp}, \mathsf{par}_a)$. The challenger first uses $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n, picks $\mathbf{v} \xleftarrow{R} \mathbb{Z}_N^n$ and $\alpha \xleftarrow{R} \mathbb{G}_1$, and sets $\mathsf{sk} = g_1^{\alpha}$. It sets the public key pk as $(g_1^{\mathbf{v}}, e(g_1, \mathsf{sk}))$ and the authority secret key msk as $(\mathbf{v}, \mathsf{sk})$. It sends pk to the adversary and adds a to the set I.

Key Queries Upon receiving a key query $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$ for an uncorrupted authority $a \in I$, $\mathcal{B}$ answers the query depending on the number distinct ID that have been queried before. If ID is one of the $(j-1)$th first IDs being queried, $\mathcal{B}$ answers with a semi-functional key of type II by sending

$\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; h_1 h_3, u_{\mathsf{ID}})$. If the query is for the jth ID, $\mathcal{B}$ answers by sending $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; T, u_{\mathsf{ID}})$. Otherwise, $\mathcal{B}$ answers with a pseudo normal key by sending $\overline{\mathsf{KeyGen}}(\mathsf{msk}_a, y; g_1, u_{\mathsf{ID}})$.

Note that all cases the key queries are answered with elements from the hash oracle's range, creating properly distributed semi-functional keys. Also, observe that if $T \in_R \mathbb{G}_{12}$, a query for the jth ID is answered with a semi-functional key of type I, and otherwise, if $T \in_R \mathbb{G}_{13}$, the query is answered with a semi-functional key of type II.

Challenge Ciphertext Whenever $\mathcal{A}$ requests the ciphertext challenge by sending $(m_0, m_1, \{x_a^*\}_{a \in \mathcal{A}^*})$, the challenger $\mathcal{B}$ picks $b \xleftarrow{R} \{0,1\}$ and encrypts message m_b as a challenge ciphertext using g_1 and $h_2' h_3'$.

Choose an $a' \in \mathcal{A}^*$, pick $\omega'_{a,1} \xleftarrow{R} \mathbb{Z}_N$ for each authority $a \in \mathcal{A}^* \setminus a'$, and set $\omega'_{a',1} = -\sum_{a \in \mathcal{A}^* \setminus a'} \omega'_{a,1}$. Additionally, pick $\omega'_{a,23}, \delta_a \xleftarrow{R} \mathbb{Z}_N$, and set $e(g_1,g_1)^{\delta_a}$ for all $a \in \mathcal{A}^*$, and define $e(g_1,g_1)^{\Delta} = \prod_{a \in \mathcal{A}^*} e(g_1,g_1)^{\delta_a}$. Blind the message $m_b \in \mathbb{G}_T$ using $e(g_1,g_1)^{\Delta}$ to obtain $\mathsf{ct}_0 = m_b \cdot e(g_1,g_1)^{\Delta}$.

Now, for each authority $a \in \mathcal{A}^*$ continue as follows (we frequently drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(N, x)$ to obtain w_1, w_2, and polynomials $(c_1, \ldots, c_{w_3})$.

If $a \in I$, pick $s_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1 + w_2]^+$, and set $\mathsf{ct}_{a,1,i} = (g_1 h_2' h_3')^{s_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = \left((g_1)^{\omega'_{a,1}} (h_2' h_3')^{\omega'_{a,23}}\right)^{\eta_\ell} \cdot \prod_{z \in [w_2]} (g_1 h_2' h_3')^{\eta_{\ell,z} s_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} (g_1 h_2' h_3')^{\eta_{\ell,i,j} s_{a,i} v_j}.$$

Blind the value $e(g_1,g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1,g_1)^{\delta_a} \cdot e(g_1^{s_{a,0}}, g_1^{\alpha_a})$.

If $a \in \bar{I}$, pick $s_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1 + w_2]^+$, and set $\mathsf{ct}_{a,1,i} = g_1^{s_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = \left((g_1)^{\omega'_{a,1}} (h_2' h_3')^{\omega'_{a,23}}\right)^{\eta_\ell} \cdot \prod_{z \in [w_2]} g_1^{\eta_{\ell,z} s_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} \left(g_1^{v_j}\right)^{\eta_{\ell,i,j} s_{a,i}}.$$

Blind the value $e(g_1,g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1,g_1)^{\delta_a} \cdot e(g_1, \mathsf{sk}_a)^{s_{a,0}}$.

The complete challenge ciphertext is

$$\mathsf{ct} = \left(\mathsf{ct}_0, \{\mathsf{ct}_{a,0}, \mathsf{ct}_{a,1,0}, \ldots, \mathsf{ct}_{a,1,w_1}, \mathsf{ct}_{a,2,1}, \ldots, \mathsf{ct}_{a,2,w_3}\}_{a \in \mathcal{A}^*}\right).$$

To see that this is properly distributed as a semi-functional ciphertext, first observe that $\omega'_{a,23} \pmod{p_2}$ is independent of $\omega'_{a,23} \pmod{p_3}$. Moreover, note that (for all i) the values $s_{a,i} \pmod{p_1}$, $s_{a,i} \pmod{p_2}$, and $s_{a,i}$

$(\text{mod } p_3)$ are mutually independent. So, the given ciphertext is distributed as a semi-functional one, and thus, we are left to prove that adversary $\mathcal{A}$ cannot distinguish a nominally semi-functional ciphertext (with $\mathcal{C} = \{1, 2\}$) from a semi-functional ciphertext (with $\mathcal{C} = \{1\}$).

Let $a' \in \mathcal{A}^* \cap I$ be an authority for which $\mathcal{A}$ cannot decrypt the ciphertext component $\mathsf{ct}_{a',0}$ because $P_{a'}(x^*_{a'}, y_{a'}) = \text{FALSE}$. Such an authority exists, as otherwise $\mathcal{B}$ would have aborted the game or $\mathcal{A}$ would have been able to trivially decrypt the challenge ciphertext. Now, observe that all values $\omega'_{a,23}$ $(\text{mod } p_2)$ look random for $a \in \mathcal{A}^* \setminus a'$, while $\omega'_{a',23} \in_R \mathbb{Z}_N \ (\text{mod } p_2)$ for semi-functional ciphertext and $\omega'_{a',23} = -\sum_{a \in \mathcal{A}^* \setminus a'} \omega'_{a,23} \ (\text{mod } p_2)$ for nominally semi-functional ciphertext. (In both nominally semi-functional and semi-functional ciphertext, all values $\omega'_{a,23} \ (\text{mod } p_3)$ for $a \in \mathcal{A}^*$, are already random.) Hence, $\mathcal{A}$'s view can *at most* contain information about $\omega'_{a,23}$ $(\text{mod } p_2)$ on the values $\{\mathbf{s}_{a'}, \mathbf{c}_{a'}(0, \mathbf{s}_{a'}, \hat{\mathbf{s}}_{a'}, \mathbf{b}_{a'}), \mathbf{r}_{a'}, \mathbf{k}_{a'}(0, \mathbf{r}_{a'}, \hat{\mathbf{r}}_{a'}, \mathbf{b}_{a'})\}$ in the subgroup $\mathbb{G}_2$ (remember, $P_{a'}(x^*_{a'}, y_{a'}) = \text{FALSE}$ for the $y_{a'}$ of the jth ID). No other information about the values in these subgroups is given by any of the key query responses (note $\mathbf{b}_{a'}$ is independent of $\mathbf{b}_a$). By the statistical security property (see Definition 14), we know that this view is now indistinguishable from $\{\mathbf{s}_{a'}, \mathbf{c}_{a'}(\omega_{a'}, \mathbf{s}_{a'}, \hat{\mathbf{s}}_{a'}, \mathbf{b}_{a'}), \mathbf{r}_{a'}, \mathbf{k}_{a'}(0, \mathbf{r}_{a'}, \hat{\mathbf{r}}_{a'}, \mathbf{b}_{a'})\}$, the view corresponding to a semi-functional ciphertext. Hence, the ciphertext is distributed correctly according to the adversary's view. Moreover, depending on the value of T, $\mathcal{B}$ either plays $\mathsf{Game}_{2,j,1}$ or $\mathsf{Game}_{2,j,2}$. □

Lemma 6 ($\mathsf{Game}_{2,q,2} \approx_c \mathsf{Game}_3$)**.** *Any p.p.t. adversary $\mathcal{A}$, making at most q key queries for distinct* IDs *and having at most a negligible advantage in breaking Assumption 7, has at most a negligible advantage in distinguishing $\mathsf{Game}_{2,q,2}$ from Game_3.*

Proof. Note that in $\mathsf{Game}_{2,q,2}$, the challenge ciphertext is semi-functional and *all* key queries are answered with a semi-functional key of type II. We have to prove that the adversary $\mathcal{A}$ cannot distinguish whether, for some $a \in \mathcal{A}$, $\mathsf{ct}_{a,0}$ is replaced by a random element in $\mathbb{Z}_N$ or not.

The challenger $\mathcal{B}$ receives $(\text{GP}, g_1, g_2, g_3, g_1^a, (g_1 g_3)^b, g_1^c, g_1^{ac} g_3^d)$ and T, where either $T = e(g_1, g_1)^{abc}$ or $T \in_R \mathbb{G}_T$. Now, $\mathcal{B}$ plays the game with $\mathcal{A}$ as follows.

Hash Oracle Upon receiving oracle query ID for the hash function H, the challenger $\mathcal{B}$ checks if it received the query before, and if so, answers with the same reply as before. If $\mathcal{A}$ has not queried for the hash value of ID before, $\mathcal{B}$ picks a value $u_{\mathsf{ID}} \xleftarrow{R} \mathbb{Z}_N$. It answers the query with $B^{-1}(g_1 g_3)^{u_{\mathsf{ID}}} = (g_1 g_3)^{-b + u_{\mathsf{ID}}}$.

Setup The challenger $\mathcal{B}$ sets $\mathsf{pp} = (\mathrm{GP}, g_1)$ and sends pp to the adversary $\mathcal{A}$.

Authority Queries Request for a new authority a using par_a are answered by the challenger by running $\mathsf{AuthoritySetup}(\mathsf{pp}, \mathsf{par}_a)$. The challenger first uses $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n, picks $\mathbf{v} \xleftarrow{R} \mathbb{Z}_N^n$ and $\tilde{\alpha} \xleftarrow{R} \mathbb{Z}_N$, and sets the public key pk as $(g_1^{a+\tilde{v}_1}, g_1^{v_2}, \ldots, g_1^{v_n}, e(g_1^a, (g_1 g_3)^b) e(g_1, g_1)^{\tilde{\alpha}})$ and (thereby indirectly) setting the authority secret key $\mathsf{msk} = (v_1 = a + \tilde{v}_1, v_2, \ldots, v_n, g_1^{ab+\tilde{\alpha}})$. It sends pk to the adversary and adds a to the set I.

Key Queries Upon receiving a key query $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$ for an uncorrupted authority $a \in I$, $\mathcal{B}$ answers the query with a semi-functional key of type II. The challenger $\mathcal{B}$ computes $\overline{\mathsf{KeyGen}}(\mathsf{sk}_a, y; g_1 g_3, u_{\mathsf{ID}})$ as follows. First, it sets $\mathsf{usk}_{a,1,0} = (g_1 g_3)^{-b+u_{\mathsf{ID}}}$ and $\mathsf{usk}_{a,1,i} = (g_1 g_3)^{r_i}$. Next, to construct the values $\mathsf{usk}_{a,2,\ell}$, consider two cases. Either k_ℓ contains both the symbol α and $b_1 r_0$, or it does not contain this combination (*i.e.*, $\phi_\ell = \phi_{\ell,0,1}$, see Section 5.4.1; symbols b_1 and r_0 may occur separately, but not in the combination $b_1 r_0$). In the case that α and $b_1 r_0$ do not occur in k_ℓ, $\mathcal{B}$ can create $\mathsf{usk}_{a,2,\ell}$ using the values $\mathsf{usk}_{a,1,0}$ and $r_1, \ldots, r_{m_2}$; and $g_1^{a+\tilde{v}_1} g_3^{\tilde{v}_1}$ and $v_2, \ldots, v_n$ (and, of course, the values ϕ_ℓ, $\phi_{\ell,z}$, and $\phi_{\ell,i,j}$). In the case that both α and $b_1 r_0$ occur in k_ℓ, observe that $\mathcal{B}$ needs to compute $(g_1 g_3)^{\phi_\ell \alpha + \sum_{z \in [m_2]} \phi_{\ell,z} \hat{r}_z + \sum_{i \in [m_1]^+, j \in [n]} \phi_{\ell,i,j} r_i b_j}$, where we have that

$$\begin{aligned} g_1^{\phi_\ell \alpha + \phi_{\ell,0,1} r_0 b_1} &= g_1^{\phi_\ell (ab+\tilde{\alpha}) + \phi_{\ell,0,1}(-b+u_{\mathsf{ID}})(a+\tilde{v}_1)} \\ &= g_1^{\phi_\ell \left((ab+\tilde{\alpha}) + (-b+u_{\mathsf{ID}})(a+\tilde{v}_1) \right)} && \text{(since, } \phi_\ell = \phi_{\ell,0,1}\text{)} \\ &= g_1^{\phi_\ell (\tilde{\alpha} - b\tilde{v}_1 + a u_{\mathsf{ID}} + \tilde{v}_1 u_{\mathsf{ID}})}. \end{aligned}$$

And so it sets (we slightly abuse notation and write $(g_1 g_3)^{v_1}$ for $(g_1^a)^{\tilde{v}_1} (g_3)^{\tilde{v}_1}$)

$$\mathsf{usk}_{a,2,\ell} = \left(g_1^{\tilde{\alpha}} \left((g_1 g_3)^b \right)^{-\tilde{v}_1} (g_1^a)^{u_{\mathsf{ID}}} (g_1 g_3)^{\tilde{v}_1 u_{\mathsf{ID}}} \right)^{\phi_\ell} \cdot \prod_{z \in [m_2]} (g_1 g_3)^{\phi_{\ell,z} r_{m_1+z}} \cdot \prod_{\substack{i \in [m_1]^+, j \in [n], \\ (i,j) \neq (0,1)}} (g_1 g_3)^{\phi_{\ell,i,j} r_i v_j}.$$

Note that the key queries are answered with elements from the hash oracle's range and create properly distributed semi-functional keys of type II.

Challenge Ciphertext Whenever $\mathcal{A}$ requests the ciphertext challenge by sending $(m_0, m_1, \{x_a^*\}_{a \in \mathcal{A}^*})$, the challenger $\mathcal{B}$ picks $b \xleftarrow{R} \{0,1\}$ and encrypts message m_b as a semi-functional challenge ciphertext.

Choose an uncorrupted authority $a' \in \mathcal{A}^* \cap I$. For each authority $a \in \mathcal{A}^* \setminus a'$, pick $\omega'_{a,1}, \delta_a \xleftarrow{R} \mathbb{Z}_N$, and set $\omega'_{a',1} = -\sum_{a \in \mathcal{A}^* \setminus a'} \omega'_{a,1}$ and *indirectly* set $\delta_{a'} = abc - \sum_{a \in \mathcal{A}^* \setminus a'} \delta_a$. Additionally, pick $\omega'_{a,23} \xleftarrow{R} \mathbb{Z}_N$ for all $a \in \mathcal{A}^*$. Blind the message $m_b \in \mathbb{G}_T$ using T to obtain $\mathsf{ct}_0 = m_b \cdot T$. Note that if $T = e(g_1, g_1)^{abc}$, the challenger simulates $\mathsf{Game}_{2,q,2}$ using $\Delta = abc$ and otherwise, if $T \in_R \mathbb{G}_T$, the challenger simulates Game_3.

Now, for each authority $a \in \mathcal{A}^*$ continue as follows (we frequently drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(N, x)$ to obtain w_1, w_2, and polynomials $(c_1, \ldots, c_{w_3})$.

If $a = a'$, pick $\tilde{s}_{a',0} \xleftarrow{R} \mathbb{Z}_N$ and $s_{a',k} \xleftarrow{R} \mathbb{Z}_N$ for $k \in [w_1 + w_2]$. Set $\mathsf{ct}_{a',1,0} = (g_1^c)^{-1}(g_1 g_2 g_3)^{\tilde{s}_{a',0}}$ and $\mathsf{ct}_{a',1,i} = (g_1 g_2 g_3)^{s_{a',i}}$ for $i \in [w_1]$. Next, $\mathcal{B}$ constructs the values $\mathsf{ct}_{a',2,\ell}$. The challenger $\mathcal{B}$ needs to compute (among others)

$$g_1^{\eta_\ell \omega + \sum_{z \in [w_2]} \eta_{\ell,z} \hat{s}_{a',z} + \sum_{i \in [w_1]^+, j \in [n]} \eta_{\ell,i,j} s_i b_j},$$

where the occurance of $s_0 b_1$ in c_ℓ can be computed by

$$\begin{aligned} g_1^{\eta_{\ell,0,1} s_0 b_1} &= g_1^{\eta_{\ell,0,1}(-c + \tilde{s}_{a',0})(a + \tilde{v}_1)} \\ &= \left((g_1^{ac})^{-1} (g_1^c)^{-\tilde{v}_1} (g_1^a)^{\tilde{s}_{a',0}} (g_1)^{\tilde{s}_{a',0}\tilde{v}_1} \right)^{\eta_{\ell,0,1}}. \end{aligned}$$

So, the challenger $\mathcal{B}$ sets (we slightly abuse notation and write $(g_1 g_2 g_3)^{v_1}$ for $(g_1^a)^{\tilde{v}_1}(g_2 g_3)^{\tilde{v}_1}$ and $(g_1 g_2 g_3)^{s_{a',0}}$ for $(g_1^c)^{-1}(g_1 g_2 g_3)^{\tilde{s}_{a',0}}$)

$$\begin{aligned} \mathsf{ct}_{a',2,\ell} = &\left((g_1)^{\omega'_{a,1}} (g_2 g_3)^{\omega'_{a,23}} \right)^{\eta_\ell} \cdot \prod_{z \in [w_2]} (g_1 g_2 g_3)^{\eta_{\ell,z} s_{a,w_1+z}} \\ &\cdot \left((g_1^{ac} g_3^d)^{-1} (g_1^c)^{-\tilde{v}_1} (g_1^a)^{\tilde{s}_{a',0}} (g_1 g_2 g_3)^{\tilde{s}_{a',0}\tilde{v}_1} \right)^{\eta_{\ell,0,1}} \\ &\cdot \prod_{\substack{i \in [w_1]^+, j \in [n], \\ (i,j) \neq (0,1)}} (g_1 g_2 g_3)^{\eta_{\ell,i,j} s_{a',i} v_j}. \end{aligned}$$

Note that by using this, $\mathcal{B}$ indirectly uses $(\omega'_{a',23} - d \cdot \eta_{\ell,0,1}/\eta_\ell)$ in subgroup $\mathbb{G}_3$ instead of $\omega'_{a',23}$. However, since $\omega'_{a',23} \in_R \mathbb{Z}_N$ and no constraint is imposed on the sum $\sum_{a \in \mathcal{A}^*} \omega'_{a',23}$, the distribution of the ciphertext component is identical to a semi-functional ciphertext.

Blind the value $e(g_1, g_1)^{\delta_{a'}}$ by setting

$$\begin{aligned} \mathsf{ct}_{a',0} &= e(g_1, g_1)^{\delta_{a'}} \cdot e(g_1, g_1)^{\alpha_{a'} s_{a',0}} \\ &= e(g_1, g_1)^{\left(abc - \sum_{a \in \mathcal{A}^* \setminus a'} \delta_a \right) + (ab + \tilde{\alpha}_{a'})(-c + \tilde{s}_{a',0})} \end{aligned}$$

$$= e(g_1, g_1)^{\left(-\sum_{a \in \mathcal{A}^* \setminus a'} \delta_a\right) + ab\tilde{s}_{a',0} - c\tilde{\alpha}_{a'} + \tilde{\alpha}_{a'}\tilde{s}_{a',0}}$$
$$= e(g_1, g_1)^{\left(-\sum_{a \in \mathcal{A}^* \setminus a'} \delta_a\right) + \tilde{\alpha}_{a'}\tilde{s}_{a',0}} e\big(g_1^a, (g_1 g_3)^b\big)^{\tilde{s}_{a',0}} e(g_1^c, g_1)^{-\tilde{\alpha}_{a'}}.$$

If $a \neq a'$, but $a \in I$, pick $s_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1 + w_2]^+$, and set $\mathsf{ct}_{a,1,i} = (g_1 g_2 g_3)^{s_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set (we slightly abuse notation and write $(g_1 g_2 g_3)^{v_1}$ for $(g_1^a)^{\tilde{v}_1}(g_2 g_3)^{\tilde{v}_1}$)

$$\mathsf{ct}_{a,2,\ell} = \Big((g_1)^{\omega'_{a,1}}(g_2 g_3)^{\omega'_{a,23}}\Big)^{\eta_\ell} \cdot \prod_{z \in [w_2]} (g_1 g_2 g_3)^{\eta_{\ell,z} s_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} \big((g_1 g_2 g_3)^{v_j}\big)^{\eta_{\ell,i,j} s_{a,i}}.$$

Blind the value $e(g_1, g_1)^{\delta_a}$ by setting

$$\mathsf{ct}_{a,0} = e(g_1, g_1)^{\delta_a} \cdot e(g_1, g_1)^{\alpha_a s_{a,0}}$$
$$= e(g_1, g_1)^{\delta_a} \cdot \big(e(g_1^a, (g_1 g_3)^b) e(g_1, g_1)^{\tilde{\alpha}_a}\big)^{s_{a,0}}.$$

If $a \in \bar{I}$, pick $s_{a,k} \in \mathbb{Z}_N$ for $k \in [w_1 + w_2]^+$, and set $\mathsf{ct}_{a,1,i} = g_1^{s_{a,i}}$ for $i \in [w_1]^+$, and for $\ell \in [w_3]$, set

$$\mathsf{ct}_{a,2,\ell} = \Big((g_1)^{\omega'_{a,1}}(g_2 g_3)^{\omega'_{a,23}}\Big)^{\eta_\ell} \cdot \prod_{z \in [w_2]} g_1^{\eta_{\ell,z} s_{a,w_1+z}} \cdot \prod_{i \in [w_1]^+, j \in [n]} \big(g_1^{v_j}\big)^{\eta_{\ell,i,j} s_{a,i}}.$$

Blind the value $e(g_1, g_1)^{\delta_a}$ by setting $\mathsf{ct}_{a,0} = e(g_1, g_1)^{\delta_a} \cdot e(g_1, \mathsf{sk}_a)^{s_{a,0}}$.

The complete challenge ciphertext is

$$\mathsf{ct} = \big(\mathsf{ct}_0, \{\mathsf{ct}_{s,0}, \mathsf{ct}_{s,1,0}, \ldots, \mathsf{ct}_{s,1,w_1}, \mathsf{ct}_{s,2,1}, \ldots, \mathsf{ct}_{s,2,w_3}\}_{s \in S}\big).$$

This semi-functional ciphertext is properly distributed, with the equality $\prod_{a \in \mathcal{A}^*} e(g_1, g_1)^{\delta_a} = e(g_1, g_1)^{abc}$. So, if $T = e(g_1, g_1)^{abc}$, the adversary $\mathcal{A}$ is playing $\mathsf{Game}_{2,q,2}$ and otherwise, if $T \in_R \mathbb{G}_T$, $\mathcal{A}$ is playing Game_3. □

Finally, note that in Game_3, the challenger gives the adversary an encryption of a random message. Hence, $\mathcal{A}$ has no advantage in winning the game.

5.7 Multi-authority Pair Encoding Examples

We give several examples of MA-PESS for various predicate families.

5.7.1 Multi-authority Identity-based Encoding

We can see the MA-ABE construction by Lewko and Waters [LW11] as a special case of our general MA-PE scheme. Their construction combines the same IBE scheme multiple times with a "multi-authority layer" on top. Based on their scheme, we extract the underlying MA-PES for an identity-based predicate. However, note that if we apply our conversion algorithm on the resulting encoding, we obtain a multi-authority IBE scheme, not an MA-ABE scheme, since our conversion uses additive secret sharing instead of Shamir secret sharing. Furthermore, the resulting MA-PES can be seen as an encoding for an IBE scheme which only allows for a *single* identity.

Example 1 (MA-PES Based On [LW11])**.** We give an MA-PES for multi-authority *identity-based encryption* based on the MA-ABE scheme by Lewko and Waters [LW11]. The pair encoding for an authority a is the following:

$$\mathbf{b} = (b_1); \mathbf{s} = (s_0); \mathbf{c} = (\omega + b_1 s_0); \mathbf{r} = (r_0); \mathbf{k} = (\alpha + b_1 r_0).$$

For Pair we have

$$E = 1, \hat{E} = -1.$$

Correctness follows by simple substitutions,

$$\begin{aligned} & s_0(E)[\alpha + b_1 r_0] + [b_1 s_0 + \omega](\hat{E}) r_0 \\ & \quad = s_0(1)[\alpha + b_1 r_0] + [b_1 s_0 + \omega](-1) r_0 \\ & \quad = s_0 \alpha + s_0 b_1 r_0 - (s_0 b_1 r_0 + \omega r_0) \\ & \quad = \alpha s_0 - \omega r_0. \end{aligned}$$

We can extend the construction to obtain a *small universe* construction for t identities, by setting

$$\mathbf{b} = (b_1, \ldots, b_t); \mathbf{s} = (s_0); \mathbf{c} = (\omega + b_{\rho(x)} s_0); \mathbf{r} = (r_0); \mathbf{k} = (\alpha + b_{\rho(y)} r_0),$$

where ρ is an injective function that maps an identity to an identity index in $[t]$.

Remark 3 (One-Use Requirement)**.** Similar to the *one-use requirement* for *attributes*, as found in several ABE schemes [LOS+10; LW11; Att14], the MA-PES of Example 1 has this one-use requirement as well, *i.e.*, a ciphertext ct from a corresponding MA-PE scheme may only contain the *identity* x, encoded by $b_{\rho(x)}$, once.

Theorem 10 (MA-PES Based On [LW11])**.** *The (extended) MA-PES described in Example 1 is statistically secure (see Definition 14).*

Proof. If $P_\kappa(x, y) = \text{FALSE}$, we have to show that the distributions

$$\{s_0, b_{\rho(x)}s_0, r_0, b_{\rho(y)}r_0\} \quad \text{and} \quad \{s_0, \omega + b_{\rho(x)}s_0, r_0, b_{\rho(y)}r_0\}$$

are statically indistinguishable, where $b_{\rho(x)}, b_{\rho(y)}, \omega, s_0, r_0 \xleftarrow{R} \mathbb{Z}_p$ for any prime p, $\log_2 p = \Theta(\lambda)$. Since $P_\kappa(x, y) = \text{FALSE}$, we know that $x \neq y$ and thus $\rho(x) \neq \rho(y)$.

We distinguish two cases:

- $s_0 \in \mathbb{Z}_p^*$, *i.e.*, s_0 is a generator of the multiplicative group $\mathbb{Z}_p^*$. Then, $b_{\rho(x)}s_0$ is uniformly distributed in $\mathbb{Z}_p$. On the other hand, $\omega + b_{\rho(x)}s_0$ is also uniformly distributed in $\mathbb{Z}_p$. Hence, the distributions are identical.

- $s_0 = 0$, *i.e.*, s_0 is not a generator for the multiplicative group $\mathbb{Z}_p^*$. Then, $b_{\rho(x)}s_0 = 0$, while $\omega + b_{\rho(x)}s_0 \in_R \mathbb{Z}_p$. However, this case only occurs with a probability negligible in λ.

Combining the two cases, we have proven that the two distributions are statistically indistinguishable. □

5.7.2 Multi-authority Attribute-based Encoding

We adapt the PES for CP-ABE from the full version of Attrapadung [Att14, Scheme 11] to an MA-PES. The PES is, in its turn, based on a *small universe* CP-ABE scheme by Lewko *et al.* [LOS+10].

Example 2 (MA-PES Based On [LOS+10; Att14])**.** The PES by Attrapadung [Att14] can be turned into an MA-PES. Let t denote the number of attributes in the universe. For a linear secret sharing scheme (LSSS) using $(\mathrm{A}^{w_3 \times w_2}, \pi)$, where we denote the ith row of A by $\mathbf{a}_i$ and π is an injective function that maps a row in A to an attribute index in $[t]$, the pair encoding for an authority a is the following:

$$\mathbf{b} = (b', b_1, \ldots, b_t);$$
$$\mathbf{s} = (s_0, s_1); c_i = (\mathbf{a}_i(\omega + s_0 b', \hat{s}_2, \ldots, \hat{s}_{w_2})^\intercal + s_1 b_{\pi(i)}) \text{ for all } i \in [w_3];$$
$$\mathbf{r} = (r_0); \mathbf{k} = (\alpha + r_0 b', \{r_0 b_y\}_y).$$

The matrices returned by the Pair algorithm are indirectly defined by the combination of keys required to satisfy the access policy as described in the ciphertext.

Correctness follows by first computing

$$\begin{aligned} & c_i \cdot r_0 - k_y \cdot s_1 \\ & \quad = [\mathbf{a}_i(\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T} + s_1 b_{\pi(i)}] r_0 - r_0 b_y \cdot s_1 \\ & \quad = \mathbf{a}_i(\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T} \cdot r_0 \qquad (\text{if } \pi(i) = y) \end{aligned}$$

for the attributes $\pi(i)$ the user has the key components $y = \pi(i)$ for. Then, if the user obtained enough shares $\mathbf{a}_i(\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T} \cdot r_0$, he can combine the shares to recover the secret $[\omega + s_0 b'] \cdot r_0$ and then use this to symbolically obtain

$$\begin{aligned} k_1 \cdot s_0 - [\omega + s_0 b'] \cdot r_0 &= [\alpha + r_0 b'] \cdot s_0 - [\omega + s_0 b'] \cdot r_0 \\ &= \alpha s_0 - \omega r_0. \end{aligned}$$

Theorem 11 (MA-PES Based On [Att14])**.** *The MA-PES described in Example 2 is statistically secure (see Definition 14).*

Proof. The proof is very similar to the proof presented in the full version of [Att14].

When $P(x, y) = \text{FALSE}$, we have that (A, π) does not accept y. We need to prove that ω is hidden. We may assume $s_1 \neq 0$ since the probability of $s_1 = 0$ is negligible in λ. For $j = 1, \dots, w_3$, we consider two cases. If $\pi(j) \notin y$, then $b_{\pi(j)}$ does not appear anywhere except for in c_i and hence the information on $\omega + s_0 b'$ will not be leaked from c_i. Now consider $\pi(j) \in y$. In this case, both s_1 and $b_{\pi(j)}$ are available (since r_0 and $r_0 b_{\pi(j)}$ are), hence $\mathbf{a}_i(\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T}$ is known. Now from the lemma of LSSS (similar to [Att14, Proposition 40]), there exists a vector $\mathbf{u} \in \mathbb{Z}_N^{w_3}$ with $u_1 \neq 0$, such that $\mathbf{u}$ is orthogonal to all $\mathbf{a}_j$, where $\pi(j) \in y$. Hence, $\mathbf{a}_j(\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T} = \mathbf{a}_j((\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T} + z\mathbf{u}^\mathsf{T})$ for any unknown random $z \in \mathbb{Z}_N$. Therefore, $\mathbf{a}_j(\omega + s_0 b', \hat{s}_2, \dots, \hat{s}_{w_2})^\mathsf{T}$ does not leak information on $\omega + s_0 b'$ as $u_1 \neq 0$. In either case $\omega + s_0 b'$ is hidden in the encoding. Since ω only occurs in this expression $\omega + s_0 b'$, no information on ω is revealed. □

5.7.3 Multi-authority Inner-product Encoding

To create a multi-authority admissible pair encoding scheme (MA-PES) for an inner-product predicate, we extend the "short secret key encoding" presented by Wee [Wee14, § 5.1]

Example 3 (MA-PES Based On [BBO4b; Wee14])**.** Based on the *predicate* encoding of Wee [Wee14] for an IPPE scheme, which, in its turn, is based on

the scheme of Boneh and Boyen [BB04b], we create an MA-PES for the inner-product predicate. Such a predicate evaluates to TRUE if and only if the inner product of the, with the ciphertext associated, vector $\mathbf{x}$ and the, with the key associated, vector $\mathbf{y}$ equals 0, *i.e.*, if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Let t be the length of the vectors $\mathbf{x}$ and $\mathbf{y}$. The pair encoding for an authority a is the following:

$$\begin{aligned}
&\mathbf{b} = (b', b'', \mathbf{b}^+), \text{ where } \mathbf{b}^+ = (b_1, \ldots, b_t);\\
&\mathbf{s} = (s_0); \mathbf{c} = (-\omega + s_0 b', s_0(b''\mathbf{x} + \mathbf{b}^+));\\
&\mathbf{r} = (r_0); \mathbf{k} = (\alpha - r_0(b' + \langle \mathbf{b}^+, \mathbf{y} \rangle)).
\end{aligned}$$

Similar to Example 2, $\mathsf{Pair}(N, x, y)$ relies on the value y, which is in this case is the vector $\mathbf{y}$. Algorithm Pair outputs matrices to compute $s_0 \cdot k_1 + \langle \mathbf{c}, (1, \mathbf{y}) \rangle \cdot r_0$.

Correctness follows by simple substitutions and simplifying the expression,

$$\begin{aligned}
&s_0 \cdot k_1 + \langle \mathbf{c}, (1, \mathbf{y}) \rangle \cdot r_0\\
&\quad = s_0 [\alpha - r_0(b' + \langle \mathbf{b}^+, \mathbf{y} \rangle)] + \langle (-\omega + s_0 b', s_0(b''\mathbf{x} + \mathbf{b}^+), (1, \mathbf{y}) \rangle r_0\\
&\quad = s_0 [\alpha - r_0(b' + \langle \mathbf{b}^+, \mathbf{y} \rangle)] + [(-\omega + s_0 b') + s_0 \langle b''\mathbf{x} + \mathbf{b}^+, \mathbf{y} \rangle] r_0\\
&\quad = s_0 \alpha - s_0 r_0 \langle \mathbf{b}^+, \mathbf{y} \rangle - \omega r_0 + s_0 [b'' \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{b}^+, \mathbf{y} \rangle] r_0\\
&\quad = \alpha s_0 - \omega r_0 \qquad (\text{if } \langle \mathbf{x}, \mathbf{y} \rangle = 0).
\end{aligned}$$

Theorem 12 (MA-PES Based On [BB04b; Wee14])**.** *The MA-PES described in Example 3 is statistically secure (see Definition 14).*

Proof. When $P(x, y) =$ FALSE, we have that $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$. We need to prove that ω is hidden. We may assume $s_0 \neq 0$ since the probability of $s_0 = 0$ is negligible in λ. Since ω only appears in c_0, we need to show that $b' s_0$ is uniformly distributed in $\mathbb{Z}_p$ and therefore no information on ω is revealed. The value b' only appears in the adversary's view elsewhere as $r_0(b' + \langle \mathbf{b}^+, \mathbf{y} \rangle)$ in k_1. Thus, we now need to show that $r_0 \langle \mathbf{b}^+, \mathbf{y} \rangle$ is statistically hidden. The value $\mathbf{b}^+$ only appears as $s_0(b''\mathbf{x} + \mathbf{b}^+)$ in the adversary's view. However, no information on the value of b'' is revealed and so, if $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$, the value $\langle \mathbf{b}^+, \mathbf{y} \rangle$ is hidden. We may conclude that b' is hidden and so is ω. □

5.8 Conclusion

We show that the concept of a multi-authority attribute-based encryption scheme can be generalized to a multi-authority predicate encryption (MA-PE) scheme for a variety of predicate families. Our generic approach allows

us to combine the best features of several predicates into a single MA-PE scheme specific to an application's needs. We achieve our result by defining a multi-authority admissible pair encoding scheme (MA-PES) and proposing a conversion technique from such an encoding to an MA-PE scheme. The obtained MA-PE schemes are decentralized, meaning that new authorities can be created without requiring any form of interaction, while no party needs to have access to a master secret. If started from statistically secure MA-PESS, the resulting MA-PE schemes are proven to be fully secure—allowing for the static corruption of authorities—in the random oracle model.

In Chapter 6, we discuss an idea for instantiating the construction in a prime-order group setting. Another direction for future work is the formalization of plaintext-privacy and predicate-privacy in MA-PES. Finally, the relation between PESS and MA-PESS is also worthwhile to study. Since the encodings are so similar, it is our impression that it is possible to devise a conversion algorithm that turns a PES into an MA-PES.

6 Directions for Extending the Work

In the chapters above, we construct several multi-client functional encryption schemes for various functionalities. In this chapter, we give detailed directions to further refine two of our constructions. First, we describe an idea for a multi-client set intersection scheme that scales linearly in the number of clients and is still secure if part of the clients are corrupted. An additional advantage of the proposed design is that the same idea could be applied to set intersection with data transfer and similar variants. As a second idea, we aim to construct multi-authority predicate encryption in prime-order groups. Since prime-order groups are generally faster than composite-order groups for a comparable security level, a prime-order construction would improve the efficiency of the schemes.

The content of this chapter is unpublished, but might provide valuable insights for constructing improved multi-client functional encryption schemes.

6.1 Towards More Efficient Corruption-Resistant Multi-client Set Intersection

In Chapter 3, we present multi-client functional encryption (MC-FE) constructions for various set operations. Our two-client constructions are fast and cannot be substantially improved in terms of efficiency. In the case of determining the cardinality of two sets, there is even no evaluation overhead when compared with the same operation on plaintext data. Our proposals for the multi-client constructions are more involved, but can still be evaluated in the order of seconds for smaller set sizes. In Section 3.7.3, we are able to construct an MC-FE scheme for determining the set intersection that is secure against corruptions, but it scales exponentially in the number of clients involved in the scheme. The same exponential complexity arose when devising an MC-FE scheme for computing the cardinality of the set intersec-

tion (see Section 3.7.1). However, for that functionality we have come up with an improved scheme that scales only linearly in the number of clients (Section 3.7.2). This latter scheme is not corruption resistant, but does seem to have the potential to make it corruption resistant. In this section, we describe an idea for creating more efficient and corruption resistant MC-FE schemes for set intersection and variants based on the intuition of the efficient multi-client cardinality scheme.

6.1.1 Intuition for MC-FE for Set Intersection Cardinality

First, let us recall the efficient construction for determining the cardinality of the intersection of three or more set using Bloom filters construction (see Section 3.7.2). We can describe the idea on an intuitive level as follows:

1. Each client encrypts both
   - the Bloom filter representation of their set, $\mathsf{BF}(\mathcal{S}_i)$; and
   - the Bloom filter representation for each element in their set,

     $$\left\{ \mathsf{BF}(x_j) \mid x_j \in \mathcal{S}_i \right\}.$$

2. The evaluator aggregates the encrypted Bloom filter representations from each client, $\mathsf{BF}(\mathcal{S}_i)$ for $1 \leq i \leq n$, resulting in an encrypted Bloom filter representation of the set intersection of the sets, $\bigcap_{i=1}^{n} \mathcal{S}_i$.

3. The evaluator picks a client i' and uses their encrypted Bloom filter representations of each element in their set, $\left\{ \mathsf{BF}(x_j) \mid x_j \in \mathcal{S}_{i'} \right\}$, to test for membership in the encrypted Bloom filter representation of the intersection.

Observe that in Step 3, the evaluator basically determines, for each set element $x_j \in \mathcal{S}_{i'}$, whether $x_j \in (\bigcap_{i=1}^{n} \mathcal{S}_i)$. By tallying how often an element is in the set intersection, we determine the cardinality of the set intersection.

There are two relevant limitations of the above described construction. Firstly, the construction only allows for the evaluation of the set membership predicate (*i.e.*, a function that merely gives a TRUE/FALSE output). Secondly, the construction is not secure against corruptions as a corrupted client has two secret shares: one to encrypt $\mathsf{BF}(\mathcal{S}_i)$ and another to encrypt $\left\{ \mathsf{BF}(x_j) \mid x_j \in \mathcal{S}_i \right\}$. Each corrupted client can use their second share to encrypt a "full" Bloom filter for an uncorrupted client, meaning that aggregation works on a strict subset of the uncorrupted clients, leaking bits of the Bloom filter representation for the intersection of this subset of uncorrupted clients.

Related Work While private membership testing for Bloom filters (*i.e.*, testing if an element x_j is represented in a Bloom filter using a private key) is theoretically solved [Ker11], no practically efficient solution is known. In the solution by Kerschbaum [Ker11], the bits in the bit string of the Bloom filter need to be encrypted c times to achieve an error rate of 2^{-c}. Moreover, the bits are encrypted using Goldwasser–Micali encryption [GM84], requiring finite fields of a large order to be secure. Even worse, to test membership, a zero-knowledge proof is required for each bit of the bit string, making this construction only of theoretical interest.

6.1.2 Extending the Idea to Determine the Set Intersection

We claim that under the assumption that a public key inner-product predicate encryption (IPPE) scheme with a key homomorphism exists, we can construct an efficient MC-FE scheme for determining the set intersection. While the overall idea is straightforward, the concrete construction might be non-trivial. Most likely, the major difficultly lies in the challenge to construct an IPPE scheme with a key homomorphism.

Inner-product Predicate for Membership Testing It is well-known that IPPE can be used for set membership testing [KSW08] (*cf.* polynomial representations of sets [KS05]), although doing so using Bloom filters might be novel (albeit a trivial variation). Let vector $\mathbf{x} \in \mathbb{Z}_p^{m+1}$ encode the bit string of a (k, m)-Bloom filter for a single set element e with x_{m+1} set to the Hamming weight of the bit string. Now, let vector $\mathbf{y} \in \mathbb{Z}_p^{m+1}$ encode the bit string of a (k, m)-Bloom filter for some set $\mathcal{S}$, where we use a random value $r \in \mathbb{Z}_p$, instead of 0 if the bit string bit is 0. We set $y_{m+1} = -1$. The inner product of the two vectors $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{m+1} x_i y_i$ equals zero if and only if $e \in \mathcal{S}$ according to the Bloom filter.

It might be easier to see how we can use IPPE for membership testing using a concrete example.

Example 4. Consider the set element e with $(3, 8)$-Bloom filter representation 00100001 (note that one hash collision has taken place). Let 01100101 be the $(3, 8)$-Bloom filter representation for some set $\mathcal{S}$.

Following the description above, we encode our set membership predicate as

$$\mathbf{x} = (\ 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2\ );$$
$$\mathbf{y} = (\ r_1 \quad r_2 \quad 1 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \quad 1 \quad -1\ ),$$

where the values r_i denote random values (*e.g.*, $r_1 \xleftarrow{R} \mathbb{Z}_p$). Observe that if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, *i.e.*, if

$$0 \cdot r_1 + 0 \cdot r_2 + 1 \cdot 1 + 0 \cdot r_4 + 0 \cdot r_5 + 0 \cdot r_6 + 0 \cdot r_7 + 1 \cdot 1 + 2 \cdot -1 = 0,$$

we know that $e \in \mathcal{S}$ as this is similar to membership testing using unencrypted Bloom filters.

Determining the Set Intersection Under Encryption To determine the set intersection, we want to secret share the IPPE decryption key using a homomorphism in the decryption key for $\mathbf{y}$. Using such a property, we could aggregate the clients' decryption keys, *e.g.*, $\sum_{i=1}^{n} \mathbf{y}_i$ in the exponent, thereby computing the set intersection in the encrypted domain. For example, if each client i holds a secret share of 1 for each bit string position $1 \leq \ell \leq m$, *i.e.*, $\sum_{i=1}^{n} \sigma_{i,\ell} = 1$ for each ℓ, we could aggregate vectors $\mathbf{y}_i$ into the final vector $\mathbf{y}$ used in the decryption key by setting

$$y_{i,\ell} = \begin{cases} r_\ell \xleftarrow{R} \mathbb{Z}_p & \text{if } \mathsf{bs}[\ell] = 0 \\ \sigma_{i,\ell} & \text{if } \mathsf{bs}[\ell] = 1. \end{cases}$$

Using IPPE for Determining the Set Intersection The crux of this solution is that we can use the ciphertext payload of the IPPE scheme, m, to encrypt the actual set element, *i.e.*, a client encrypts all the set elements using the IPPE scheme where we associate the ciphertext with the vector $\mathbf{x}$ that is based on the Bloom filter representations of that very same set element. Thus, if the membership test succeeds (*i.e.*, the predicate returns TRUE), the evaluator can also decrypt the ciphertext to learn m, the set element itself!

For security, we require that the vector $\mathbf{x}$ is hidden in the ciphertext, *i.e.*, we require *plaintext privacy*. Surprisingly enough, we can suffice with a public key IPPE scheme: Realize that we only require to hide the values for $\mathbf{x}$ (already covered by plaintext privacy), but we may leak the entire Bloom filter $\mathbf{y}$—as long as $\mathbf{y}$ doesn't leak any information on the values $\mathbf{y}_i$. To see this, recognize that the set intersection is what we want to determine and that is all what is leaked from the Bloom filter $\mathbf{y}$. We can even allow *anyone* that has access to this Bloom filter to run an exhaustive search on the filter to try to learn the intersection. It might be insightful to understand that we only provide the evaluator with the individually encrypted set elements to allow for efficiently checking only a relatively small set of potential members. Furthermore, observe that the proposed construction has the additional benefit of lowering the false positive rate of membership testing by also considering the Hamming weight of the bit string.

6.1.3 Further Extensions to Other Set Operations

Upon completion of the above described construction for determining the set intersection, it is trivial to extend to for example set intersection with data transfer/projection: Just encrypt the associated data instead of the set element. A threshold construction might also be possible to realize this way: Encrypt secret shares instead and use again a layer of secret sharing like in the two client case (see Section 3.6.4) to recover a secret key only if at least t elements were in the set.

6.2 Towards Multi-authority Predicate Encryption in Prime-Order Groups

The proposed construction in Chapter 5 is the first generic construction for multi-authority predicate encryption (MA-PE). The result is achieved by using composite-order bilinear groups. A drawback of these composite-order groups, is that they are much slower than prime-order groups for the same security level. So, if we want to apply the construction in practice, we rather use groups of prime order than of a composite one. In this section, we explain one of our attempts in constructing a generic MA-PE scheme using prime-order bilinear groups. We believe that with more research, this approach will lead to such a scheme. However, the current approach seems to lead to a construction that is only provable without authority corruptions.

6.2.1 Bird's-Eye View of the Approach

We try to minimize the changes needed to modify our in Chapter 5 proposed composite-order MA-PE construction to a prime-order one. To do so, we attempt to replace the composite-order groups of three primes by prime-order groups that can emulate them. In that way, we can keep on using the proof structure of the multi-system encryption technique (*i.e.*, the proof technique where we use semi-functional ciphertext and keys) and aim to prove the construction *fully secure* instead of the weaker *selectively secure*. The property that makes it possible to prove the construction fully secure with composite-order bilinear groups, is that we can encode multiple independent "system groups" in a single group element, where the pairing operation applies to these system groups independently. That is, $e(g^{(12)}, h^{(12)}) = e(g^{(1)}, h^{(1)})\, e(g^{(2)}, h^{(2)})$, where we denote with $g^{(1)}$, $g^{(2)}$, and $g^{(12)}$ an element $g \in \mathbb{G}$ in system group 1, 2, or in both system group 1 and 2, respectively. To prove full security, this property should additionally fulfill two other properties. First, elements should not reveal to which system groups they belong. Second, even if the randomness

in one system group is revealed, the randomness of any other system group should remain information-theoretically hidden.

As explained above, a major drawback of composite-order groups, is that they are very inefficient. Prime-order group do not suffer from this efficiency drawback, however, they do not have the property of encoding multiple system groups into a single group element. The works by Lewko [Lew12] and Chen and Wee [CW14] overcome this limitation of prime-order groups by imposing extra structure onto these groups achieving the same desired properties as composite order group have.

We introduce *k-system groups* (kSG), a generalization of dual system groups [CW14]. kSG can be seen as an abstraction of composite-order bilinear groups. With our instantiation of triple system groups (TSG) (or, 3-system groups) in a prime-order group setting, we aim to construct prime-order multi-authority predicate encryption schemes.

6.2.2 Preliminaries

We denote the identity matrix of size $n \times n$ by I_n. To sample a random matrix from the general linear group (*i.e.*, invertible matrices) of size $n \times n$ over the finite field $\mathbb{Z}_p$, we write $\mathrm{M} \xleftarrow{R} \mathrm{GL}_n(\mathbb{Z}_p)$. Similarly, to sample a $n \times n$ diagonal matrix over $\mathbb{Z}_p^*$, we write $\mathrm{U} \xleftarrow{R} \mathrm{Diag}_n(\mathbb{Z}_p^*)$

A bilinear map over matrices is defined as

$$e(g^{\mathrm{A}}, h^{\mathrm{B}}) = e(g, h)^{\mathrm{A}^\intercal \mathrm{B}}$$

6.2.3 Complexity Assumptions

We rely on a generalized d-linear game (Definition 15) similar to the generalized d-linear assumption defined in [AC15, § B.2]. Likewise to their result, we reduce the d-linear assumption (Assumption 2) to this generalized d-linear game. The generalized d-linear assumption without relying on pairing groups has been used before [Fre10; CW14].

Reduction from the d-Linear Assumption We prove a lemma that we will use in many of the other proofs, showing that an adversary capable of breaking the Generalized d-Linear Game, can also break the d-Linear Assumption.

Definition 15 (Generalized d-Linear Game (*cf.* [AC15, § B.2]))**.** Any probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ has at most a negligible advantage in winning the following left-or-right security game.

Setup Let $\mathsf{gp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)) \leftarrow \mathcal{G}_3(1^\lambda)$ be the generated group parameters and pick values $\mathbf{a} \xleftarrow{R} (\mathbb{Z}_p^*)^d$ and $a_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$. The group parameters gp and the values $(g_1^{\mathbf{a}}, g_1^{a_{d+1}}, g_2^{\mathbf{a}})$ are shared with the adversary.

The challenger picks a bit $b \xleftarrow{R} \{0,1\}$.

Challenge Query The adversary may query the challenger for a polynomial number of challenges. Upon receiving a challenge request, the challenger picks values $\mathbf{t} \xleftarrow{R} (\mathbb{Z}_p)^d$ and $t_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$. Depending on the bit b, the challenger set

$$T_0 = g_1^{a_{d+1}\sum_{i=1}^{d} t_i} \qquad \text{or} \qquad T_1 = g_1^{a_{d+1}\sum_{i=1}^{d} t_i + t_{d+1}}$$

and returns $(g_1^{\mathbf{a}\circ\mathbf{t}}, T_b)$ to the adversary.

Guess The adversary outputs its guess b' for bit b. We define the advantage of the adversary in winning the game as $\Pr[b' = b] - 1 / 2$.

Lemma 7 (d-Linear Assumption $\leq$ Generalized d-Linear Game (*cf.* [AC15, Lemma 8])). *If the d-Linear Assumption (Assumption 2) holds for a group generator $\mathcal{G}_3(1^\lambda)$, then the Generalized d-Linear Game (Definition 15) also holds for $\mathcal{G}_3(1^\lambda)$.*

Proof. We construct a challenger capable of breaking the d-Linear Assumption using a p.p.t. adversary $\mathcal{A}$ that has a non-negligible advantage in winning the Generalized d-Linear Game.

Setup Algorithm $\mathcal{B}$ obtains

$$\left(\mathsf{gp}, g_1^{\mathbf{a}}, g_1^{a_{d+1}}, g_2^{\mathbf{a}}, g_1^{\mathbf{a}\circ\mathbf{t}}, T = g_1^{a_{d+1}\sum_{i=1}^{d} t_i + t_{d+1}}\right)$$

as input, where t_{d+1} is either 0 or uniformly chosen from $\mathbb{Z}_p^*$. It shares the group parameters gp and the values $(g_1^{\mathbf{a}}, g_1^{a_{d+1}}, g_2^{\mathbf{a}})$ with the adversary.

Challenge Query The challenger $\mathcal{B}$ picks values $\mathbf{t}' \xleftarrow{R} (\mathbb{Z}_p)^d$ and $t'_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$. It indirectly sets $\tilde{\mathbf{t}} = (\mathbf{t}+\mathbf{t}')t'_{d+1}$ and $\tilde{t}_{d+1} = t_{d+1}t'_{d+1}$ by returning the values

$$g_1^{\mathbf{a}\circ\tilde{\mathbf{t}}} = \left(g_1^{\mathbf{a}\circ\mathbf{t}} \circ g_1^{\mathbf{a}\circ\mathbf{t}'}\right)^{t'_{d+1}} \qquad \text{and} \qquad \tilde{T} = \left(T \cdot g_1^{a_{d+1}\sum_{i=1}^{d} t'_i}\right)^{t'_{d+1}}$$

to the adversary $\mathcal{A}$. Note that this returned challenge is correctly distributed.

Guess Upon receiving the guess b' for b from adversary $\mathcal{A}$, the challenger guesses that $t_{d+1} = 0$ if and only if $b' = 0$. Observe that the challenger now obtains the same non-negligible advantage in breaking the d-Linear Assumption as the adversary in winning the Generalized d-Linear Game. □

Remark 4. We require the generalized d-linear game, and thus also the d-linear assumption, to hold in both groups.

We also require the d-Bilinear Diffie–Hellman assumption, which is proven to hold under the d-linear assumption [BSW13].

Assumption 8 (d-Bilinear Diffie–Hellman [BSW13])**.** Let the bilinear map parameters $\mathsf{gp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g)$ be generated by $\mathcal{G}_3(1^\lambda)$, and $g_1 \xleftarrow{R} \mathbb{G}_1$, $g_2 \xleftarrow{R} \mathbb{G}_2$, and $x, y, a_1, \ldots, a_d, z_1, \ldots, z_d \xleftarrow{R} \mathbb{Z}_p$. Given g_1, g_2, g_2^x, g_1^y, $g_1^{a_1}, \ldots, g_1^{a_d}, g_2^{a_1}, \ldots, g_2^{a_d}, g_1^{a_1 z_1}, \ldots, g_1^{a_d z_d}, g_2^{a_1 z_1}, \ldots, g_2^{a_d z_d}$, and T, it is hard to distinguish $T = e(g_1, g_2)^{xy(z_1 + \cdots + z_d)}$ from $T \in_R \mathbb{G}_T$. That is, the advantage of any p.p.t. adversary $\mathcal{A}$ in distinguishing,

$$\Big|\Pr[\mathcal{A}((\mathsf{gp}, g_1, g_2, g_2^x, g_1^y, g_1^{\mathbf{a}}, g_2^{\mathbf{a}}, g_1^{\mathbf{a}\circ\mathbf{z}}, g_2^{\mathbf{a}\circ\mathbf{z}}), e(g_1, g_2)^{xy(z_1+\cdots+z_d)}) = 1] - \Pr[\mathcal{A}((\mathsf{gp}, g_1, g_2, g_2^x, g_1^y, g_1^{\mathbf{a}}, g_2^{\mathbf{a}}, g_1^{\mathbf{a}\circ\mathbf{z}}, g_2^{\mathbf{a}\circ\mathbf{z}}), T \xleftarrow{R} \mathbb{G}_T) = 1]\Big|,$$

where we write $\mathbf{a}$ for the column vector $(a_1, \ldots, a_d)^\intercal$ and $\mathbf{z}$ for $(z_1, \ldots, z_d)^\intercal$, is negligible in the security parameter λ.

6.2.4 k-System Groups

As explained above, we choose to introduce an abstraction layer that imposes additional structure on the bilinear groups used. The most important functional requirement of this additional structure is to obtain *system groups* within a bilinear group. A key property of these system groups—termed *orthogonality*—is that if we pair an element from one of these system groups with an element of another system groups, we obtain the identity element in the target group. This is similar to the incredibly useful property found in composite-order bilinear groups.

We base our *k-system groups* on the dual system groups introduced by Chen and Wee [CW14]. A clear difference between the two notions is that our notion achieves k of these system groups, while before only systems for achieving two system groups were described. However, our extension is more than just an increase in the number of available system groups. For example, we introduce the possibility to have multiple sampling parameter within the same group parameters and define a new mapping function from an arbitrary group element to a specific system group. Additionally, we define stronger security properties on the kSG, making them a more powerful tool.

To keep track of which system group or system groups an element belongs, we write the system group numbers between parentheses in superscript to the element. For example, $g^{(1)}$ is an element belonging to the first system

group, while $h^{(12)} = h^{(1)}h^{(2)}$ belongs to both system group 1 and 2. An example of the orthogonality requirement as stated above, can now simply be denoted as $e(g^{(1)}, h^{(12)}) = e(g^{(1)}, h^{(1)}) \cdot e(g^{(1)}, h^{(2)}) = e(g^{(1)}, h^{(1)})$.

Definition 16 (k-System Groups)**.** An instantiation of k-system groups (kSG) can be described by the following six p.p.t. algorithms.

SampGroup$(1^\lambda, k) \to (\text{GP}, \text{GS})$**.** On input of the security parameter λ and the number of system groups k, the algorithm outputs the public and secret group parameters (GP, GS). The public group parameters GP contain at least the tuple $(N = \text{ord}(\mathbb{G}_T), \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathbb{G}_r)$ and the bilinear map $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$.

SampP$(\text{GP}, 1^n) \to (\mu, \text{SP}_{\mathbb{G},1}, \dots, \text{SP}_{\mathbb{G},k}, \text{SP}_{\mathbb{H},1}, \dots, \text{SP}_{\mathbb{H},k}, \text{TR})$**.** On input of the public group parameters, the algorithm outputs a linear map μ on the domain $\mathbb{H}$, k sampling parameters for both $\mathbb{G}$ and $\mathbb{H}$, $\text{SP}_{\mathbb{G},i}$ and $\text{SP}_{\mathbb{H},i}$, respectively (for $i \in [k]$), and a trapdoor TR.

The sampling parameters $\text{SP}_{\mathbb{G},i}$ and $\text{SP}_{\mathbb{H},i}$ can be used to sample elements using SampG_i and SampH_i, respectively. The sampling parameters TR can be used in MapH_1.

SampGT$(\mu(h); s)$**.** The algorithm takes an element $\mu(h)$, $h \in \mathbb{H}$, from the image of μ and randomness $s \in \mathbb{G}_r$ as input. It outputs an element in the target group $\mathbb{G}_T$.

SampG$_i(\text{SP}_{\mathbb{G},i}; s)$**.** Outputs $n+2$ elements of $\mathbb{G}$, $\mathbf{g}^{(i)} \in \mathbb{G}^{n+2}$, sampled using randomness $s \in \mathbb{G}_r$ and using $\text{SP}_{\mathbb{G},i}$. We often omit the value s, in such a case we use $s \xleftarrow{R} \mathbb{G}_r$.

SampH$_i(\text{SP}_{\mathbb{H},i}; r)$**.** Outputs $n+2$ elements of $\mathbb{H}$, $\mathbf{h}^{(i)} \in \mathbb{H}^{n+2}$, sampled using randomness $r \in \mathbb{G}_r$ and using $\text{SP}_{\mathbb{H},i}$. We often omit the value r, in such a case we use $r \xleftarrow{R} \mathbb{G}_r$.

MapH$_1(\text{TR}, h)$**.** Maps $h \in \mathbb{H}$ to $n+2$ elements in $\mathbb{H}$, $\mathbf{h}^{(1)} \in \mathbb{H}^{n+2}$, using TR.

6.2.5 Properties of k-System Groups

To be useful, instantiations of kSG have to satisfy the properties of *correctness*, *orthogonality*, *non-degeneracy*, and *indistinguishability*.

To meet correctness, the following two properties need to hold.

Projective For all $h \in \mathbb{H}$ and values $s \in \mathbb{G}_r$, $(g_0^{(1)}, \dots) \leftarrow \mathsf{SampG}_1(\text{SP}_{\mathbb{G},1}; s)$, we have $\mathsf{SampGT}(\mu(h); s) = e(g_0^{(1)}, h)$.

Associative For all sampled elements $(g_0^{(1)}, \dots, g_{n+1}^{(1)}) \leftarrow \mathsf{SampG}_1(\text{SP}_{\mathbb{G},1})$ and $(h_0^{(1)}, \dots, h_{n+1}^{(1)}) \leftarrow \mathsf{SampH}_1(\text{SP}_{\mathbb{H},1})$, we have that for all $i \in [n+1]$, $e(g_0^{(1)}, h_i^{(1)}) = e(g_i^{(1)}, h_0^{(1)})$.

For our construction of MA-PE, we also require a third, homomorphic property to meet correctness.

Homomorphic The algorithm SampG_i is homomorphic in the randomness s used, *i.e.*,

$$\mathsf{SampG}_i(\mathrm{SP}_{\mathbb{G},i}, s_1) \cdot \mathsf{SampG}_i(\mathrm{SP}_{\mathbb{G},i}, s_2) = \mathsf{SampG}_i(\mathrm{SP}_{\mathbb{G},i}, s_1 + s_2).$$

For security, several other properties need to hold.

Orthogonality $\mu(h^{(i)}) = \mathbb{1}$ for all $2 \leq i \leq k$.

Non-degeneracy For all $2 \leq i \leq k$, sample $(g_0^{(i)}, \dots) \leftarrow \mathsf{SampG}_i(\mathrm{SP}_{\mathbb{G},i})$ and $(h_0^{(i)}, \dots) \leftarrow \mathsf{SampH}_i(\mathrm{SP}_{\mathbb{H},i})$; we have that $e(g_0^{(i)}, h_0^{(i)})^a$, where $a \xleftarrow{R} \mathbb{Z}_N$, is identically distributed to the uniform distribution over $\mathbb{G}_T$.

Indistinguishability Several forms of indistinguishability need to hold. We require $(\{1\}, \{1, \dots, k\})$-system group indistinguishability to hold in $\mathbb{G}$ (see Definition 17). Moreover, for indistinguishability in $\mathbb{H}$, we require Definition 18. Additionally, two distributions as defined in parameter-hiding (Definition 19) should be identically distributed.

Definition 17 ($(\mathcal{L}, \mathcal{R})$-System Group Indistinguishability in $\mathbb{G}$)**.** An instantiation of k-system groups is $(\mathcal{L}, \mathcal{R})$-*system group indistinguishable in* $\mathbb{G}$ for the sets $\mathcal{L}, \mathcal{R} \subseteq [k]$, if every p.p.t. adversary $\mathcal{A}$ has at most a negligible advantage in winning the following game.

Setup The challenger $\mathcal{B}$ chooses a random bit b. Next, the challenger runs the SampGroup algorithm and gives GP to the adversary.

Query The adversary may query the challenger for sampling parameters or for samples in $\mathbb{G}$.

- **Parameters** The adversary may query for the qth sampling parameters and send the parameters size n to the challenger. The challenger $\mathcal{B}$ runs $\mathsf{SampP}(\mathrm{GP}, 1^n)$ to obtain $(\mu_q, \mathrm{SP}_{q,\mathbb{G},1}, \dots, \mathrm{SP}_{q,\mathbb{G},k}, \mathrm{SP}_{q,\mathbb{H},1}, \dots, \mathrm{SP}_{q,\mathbb{H},k}, \mathrm{TR})$. It gives the map μ_q and the sampling parameters $(\mathrm{SP}_{q,\mathbb{G},1}, \dots, \mathrm{SP}_{q,\mathbb{G},k})$, $\mathrm{SP}_{q,\mathbb{H},1}$, and TR_q to $\mathcal{A}$.
- **Samples in** $\mathbb{G}$ The adversary may query for samples in $\mathbb{G}$ and send a query number q to the challenger. For $i \in [k]$, the challenger $\mathcal{B}$ runs

$$\mathbf{g}^{(\mathbf{i})} = (g_0^{(i)}, \dots, g_{n+1}^{(i)}) \leftarrow \mathsf{SampG}_i(\mathrm{SP}_{q,\mathbb{G},i}).$$

Depending on the challenger's bit b, it gives the adversary

$$\begin{cases} \mathbf{g}^{(\mathcal{L})} = \left(\prod_{i\in\mathcal{L}} g_0^{(i)}, \dots, \prod_{i\in\mathcal{L}} g_{n+1}^{(i)}\right) & \text{if } b = 0, \\ \mathbf{g}^{(\mathcal{R})} = \left(\prod_{i\in\mathcal{R}} g_0^{(i)}, \dots, \prod_{i\in\mathcal{R}} g_{n+1}^{(i)}\right) & \text{if } b = 1. \end{cases}$$

Guess The adversary outputs its guess b' for the bit b. We define the advantage of the adversary in winning the game as $\Pr[b' = b] - 1 / 2$.

Definition 18 ($(\mathcal{L}, \mathcal{R}, z)$-System Group Indistinguishability in $\mathbb{H}$)**.** An instantiation of k-system groups is *$(\mathcal{L}, \mathcal{R}, z)$-system group indistinguishable in* $\mathbb{H}$ for sets $\mathcal{L}, \mathcal{R} \subseteq [k]$ and $z \in \mathcal{L} \cap \mathcal{R}$, if every p.p.t. adversary $\mathcal{A}$ has at most a negligible advantage in winning the following game.

Setup The challenger $\mathcal{B}$ chooses a random bit b. Next, the challenger runs the $\mathsf{SampGroup}$ algorithm and gives GP to the adversary.

Query The adversary may query the challenger for sampling parameters, or for samples in $\mathbb{G}$ or $\mathbb{H}$.

- **Parameters** The adversary may query for the qth sampling parameters and send the parameters size n to the challenger. The challenger $\mathcal{B}$ runs $\mathsf{SampP}(\mathrm{GP}, 1^n)$ to obtain $(\mu_q, \mathrm{SP}_{q,\mathbb{G},1}, \dots, \mathrm{SP}_{q,\mathbb{G},k}, \mathrm{SP}_{q,\mathbb{H},1}, \dots, \mathrm{SP}_{q,\mathbb{H},k}, \mathrm{TR})$. It gives the map μ_q and the sampling parameters $\mathrm{SP}_{q,\mathbb{G},1}$, $\mathrm{SP}_{q,\mathbb{G},i}$ for $i \in (\mathcal{L} \setminus \{z\}) \cup ([k] \setminus \mathcal{R})$, and $(\mathrm{SP}_{q,\mathbb{H},1}, \dots, \mathrm{SP}_{q,\mathbb{H},k})$ to the adversary. Notice that the adversary does not receive the trapdoor TR.

- **Samples in $\mathbb{G}$** The adversary may query for samples in $\mathbb{G}$ and send a query number q along with a value $j \in [k] \setminus \{z\}$ to the challenger. The challenger $\mathcal{B}$ runs for $i \in \{z, j\}$

  $$\mathbf{g}^{(\mathbf{i})} = (g_0^{(i)}, \dots, g_n^{(i)}) \leftarrow \mathsf{SampG}_i(\mathrm{SP}_{q,\mathbb{G},i})$$

  and sends $\mathbf{g}^{(\mathbf{z},\mathbf{j})} = (g_0^{(z)} g_0^{(j)}, \dots, g_n^{(z)} g_n^{(j)})$ to the adversary.

- **Samples in $\mathbb{H}$** The adversary may query for samples in $\mathbb{H}$ and send a query number q to the challenger. For $i \in [k]$, the challenger $\mathcal{B}$ runs

  $$\mathbf{h}^{(\mathbf{i})} = (h_0^{(i)}, \dots, h_n^{(i)}) \leftarrow \mathsf{SampH}_i(\mathrm{SP}_{q,\mathbb{H},i}).$$

  Depending on the challenger's bit b, it gives the adversary

  $$\begin{cases} \mathbf{h}^{(\mathcal{L})} = \left(\prod_{i\in\mathcal{L}} h_0^{(i)}, \dots, \prod_{i\in\mathcal{L}} h_n^{(i)}\right) & \text{if } b = 0, \\ \mathbf{h}^{(\mathcal{R})} = \left(\prod_{i\in\mathcal{R}} h_0^{(i)}, \dots, \prod_{i\in\mathcal{R}} h_n^{(i)}\right) & \text{if } b = 1. \end{cases}$$

Guess The adversary outputs its guess b' for the bit b. We define the advantage of the adversary in winning the game as $\Pr[b' = b] - 1 / 2$.

Definition 19 (Parameter-hiding)**.** An instantiation of k-system groups is *parameter-hiding* in system group i, if the following two experiments for $b \in \{0, 1\}$ are perfectly indistinguishable (*i.e.*, their distributions are identical).

Setup The challenger $\mathcal{B}$ runs the SampGroup algorithm and gives GP to the adversary.

Query The adversary may query the challenger for sampling parameters, or for samples in $\mathbb{G}$ or $\mathbb{H}$.

- **Parameters** The adversary may query for the qth sampling parameters and send the parameters size n to the challenger. The challenger $\mathcal{B}$ runs $\mathsf{SampP}(\mathrm{GP}, 1^n)$ to obtain $(\mu_q, \mathrm{SP}_{q,\mathbb{G},1}, \dots, \mathrm{SP}_{q,\mathbb{G},k}, \mathrm{SP}_{q,\mathbb{H},1}, \dots, \mathrm{SP}_{q,\mathbb{H},k}, \mathrm{TR})$. Additionally, it picks values $\gamma_{q,1}, \dots, \gamma_{q,n+1}$ which will be used to uniformly re-randomize a value $g^{(i)}$ of system group i. It gives the map μ_q and the sampling parameters $\mathrm{SP}_{q,\mathbb{G},1}$ and $\mathrm{SP}_{q,\mathbb{H},1}$ to the adversary.

- **Samples in $\mathbb{G}$** The adversary may query for samples in $\mathbb{G}$ and send a query number q to the challenger. The challenger $\mathcal{B}$ runs

  $$\mathbf{g}^{(\mathbf{i})} = (g_0^{(i)}, \dots, g_{n+1}^{(i)}) \leftarrow \mathsf{SampG}_i(\mathrm{SP}_{q,\mathbb{G},i}).$$

  Depending on the challenger's bit b, it gives the adversary

  $$\begin{cases} \mathbf{g}^{(\mathbf{i})} = (g_0^{(i)}, \dots, g_{n+1}^{(i)}) & \text{if } b = 0, \\ \mathbf{g}^{(\mathbf{i})} \circ \mathbf{g}^{(\mathbf{i})\prime} = \left(g_0^{(i)}, g_1^{(i)} {g_0^{(i)}}^{\gamma_{q,1}}, \dots, g_{n+1}^{(i)} {g_0^{(i)}}^{\gamma_{q,n+1}}\right) & \text{if } b = 1. \end{cases}$$

- **Samples in $\mathbb{H}$** The adversary may query for samples in $\mathbb{H}$ and send a query number q to the challenger. The challenger $\mathcal{B}$ runs

  $$\mathbf{h}^{(\mathbf{i})} = (h_0^{(i)}, \dots, h_{n+1}^{(i)}) \leftarrow \mathsf{SampH}_i(\mathrm{SP}_{q,\mathbb{H},i}).$$

  Depending on the challenger's bit b, it gives the adversary

  $$\begin{cases} \mathbf{h}^{(\mathbf{i})} = (h_0^{(i)}, \dots, h_{n+1}^{(i)}) & \text{if } b = 0, \\ \mathbf{h}^{(\mathbf{i})} \circ \mathbf{h}^{(\mathbf{i})\prime} = \left(h_0^{(i)}, h_1^{(i)} {h_0^{(i)}}^{\gamma_{q,1}}, \dots, h_{n+1}^{(i)} {h_0^{(i)}}^{\gamma_{q,n+1}}\right) & \text{if } b = 1. \end{cases}$$

6.2.6 Construction for Prime-Order Triple System Groups

We give a construction for a prime-order TSG using prime-order groups with a Type 3 bilinear map defined on them. To ease the notation, we introduce two projection maps $\pi_i(\cdot)$ and $\rho_i(\cdot)$. Let $\pi_i(\cdot)$, for $i \in [3]$, be a projection map that maps a $3d \times 3d$ matrix to the $3d \times d$ submatrix consisting of the d columns $d(i-1)+1$ till di. Similarly, the projection map $\rho_i(\cdot)$ is defined as the map from a $3d \times 3d$ matrix to the $d \times 3d$ submatrix consisting of the d rows $d(i-1)+1$ till di. We extend the notation (by slightly abusing it) to denote the ith number of d rows of a submatrix or vector $\mathbf{v}$ as $\rho_i(\mathbf{v})$ and similarly $\pi_i(\mathbf{v})$ for a column vector.

Without showing the details here, we claim that it is easy to extend this instantiation of prime-order TSG to prime-order kSG for any constant k.

SampGroup$(1^\lambda, 3)$**.** Define the elements

$$
\begin{aligned}
&(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}_3(1^\lambda)\\
&(\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e) = (\mathbb{G}_1^{3d}, \mathbb{G}_2^{3d}, \mathbb{G}_T, e)\\
&\mathrm{U} \xleftarrow{R} \mathrm{Diag}_{3d}(\mathbb{Z}_p^*) \qquad \text{with } \mathrm{I}_{2d} \text{ in the lower-right corner,}
\end{aligned}
$$

where $d \geq 2$ is determined by the security parameter λ. Note that a single TSG element is represented by $3d$ group elements in $\mathbb{G}$ and $\mathbb{H}$.

The global public group parameters GP are $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathbb{G}_r = (\mathbb{Z}_p)^{3d}, e, g_1, g_2, g_1^{\mathrm{U}})$. The global secret group parameters GS are empty in this prime order construction.

SampP$(\mathrm{GP}, 1^n)$**.** Define

$$
\begin{aligned}
&\mathrm{B} \xleftarrow{R} \mathrm{GL}_{3d}(\mathbb{Z}_p)\,, \mathrm{B}^* = (\mathrm{B}^{-1})^\intercal;\\
&\mathrm{A}_0, \ldots, \mathrm{A}_n \xleftarrow{R} (\mathbb{Z}_p)^{3d \times 3d}
\end{aligned}
$$

and the linear map

$$
\mu \colon \mathbf{h} \mapsto e\Big(g_1^{\pi_1(\mathrm{BU})}, \mathbf{h}\Big) \qquad \text{for all } \mathbf{h} \in \mathbb{H}.
$$

Note that we can compute g_1^{BU} from $g_1^{\mathrm{U}} \in \mathrm{GP}$ and the matrix B. Output the linear map μ together with the sampling parameters for $i \in [3]$,

$$
\begin{aligned}
\mathrm{SP}_{\mathbb{G},i} &= \Big(g_1^{\pi_i(\mathrm{BU})}, g_1^{\pi_i(\mathrm{BA}_0\mathrm{U})}, \ldots, g_1^{\pi_i(\mathrm{BA}_n\mathrm{U})}\Big)\,;\\
\mathrm{SP}_{\mathbb{H},i} &= \Big(g_2^{\pi_i(\mathrm{B}^*)}, g_2^{\pi_i(\mathrm{B}^*\mathrm{A}_0^\intercal)}, \ldots, g_2^{\pi_i(\mathrm{B}^*\mathrm{A}_n^\intercal)}\Big)\,;\\
\mathrm{TR} &= (\pi_1(\mathrm{B}^*), \pi_1(\mathrm{B}^*\mathrm{A}_0^\intercal), \ldots, \pi_1(\mathrm{B}^*\mathrm{A}_n^\intercal))\,.
\end{aligned}
$$

SampGT($\mu(\mathbf{h});\mathbf{s}$). Using $\mathbf{s} \xleftarrow{R} (\mathbb{Z}_p)^{3d}$, output $g_T^{\rho_1(\mathbf{s})^\intercal \mathbf{k}}$, where $g_T^{\mathbf{k}} = \mu(\mathbf{h})$.

SampG$_i$(SP$_{\mathbb{G},i}$;$\mathbf{s}$). Using $\mathbf{s} \xleftarrow{R} (\mathbb{Z}_p)^{3d}$, output $n+2$ samples

$$\left(g_1^{\pi_i(\mathrm{BU})\rho_i(\mathbf{s})}, g_1^{\pi_i(\mathrm{BA}_0\mathrm{U})\rho_i(\mathbf{s})}, \dots, g_1^{\pi_1(\mathrm{BA}_n\mathrm{U})\rho_i(\mathbf{s})}\right).$$

SampH$_i$(SP$_{\mathbb{H},i}$;$\mathbf{r}$). Using $\mathbf{r} \xleftarrow{R} (\mathbb{Z}_p)^{3d}$, output $n+2$ samples

$$\left(g_2^{\pi_i(\mathrm{B}^*)\rho_i(\mathbf{r})}, g_2^{\pi_i(\mathrm{B}^*\mathrm{A}_0^\intercal)\rho_i(\mathbf{r})}, \dots, g_2^{\pi_i(\mathrm{B}^*\mathrm{A}_n^\intercal)\rho_i(\mathbf{r})}\right).$$

MapH$_1$(SP$_{\mathbb{H},1}, g_2^{\mathbf{r}}$). On input of $g_2^{\mathbf{r}} \in \mathbb{H}$, use $g_2^{\rho_1(\mathbf{r})} \in (\mathbb{G}_2)^d \subset \mathbb{H}$ to output the $n+2$ elements in $\mathbb{H}$,

$$\left(g_2^{\pi_1(\mathrm{B}^*)\rho_1(\mathbf{r})}, g_2^{\pi_1(\mathrm{B}^*\mathrm{A}_0^\intercal)\rho_1(\mathbf{r})}, \dots, g_2^{\pi_1(\mathrm{B}^*\mathrm{A}_n^\intercal)\rho_1(\mathbf{r})}\right).$$

Observe that we can compute these values since we know the values $\pi_1(\mathrm{B}^*)$, $\pi_1(\mathrm{B}^*\mathrm{A}_0^\intercal), \dots, \pi_1(\mathrm{B}^*\mathrm{A}_n^\intercal)$ from TR.

6.2.7 Proofs for the Properties of Our TSG Construction

We prove that the construction from Section 6.2.6 satisfies the properties that an instance of TSG needs to satisfy.

Theorem 13. *The prime-order triple system groups construction is correct.*

Proof. We can directly prove the projective property via

$$\begin{aligned}\mathsf{SampGT}(\mu(\mathbf{h});\mathbf{s}) &= e\left(g_1^{\pi_1(\mathrm{BU})\rho_1(\mathbf{s})}, \mathbf{h}\right)\\ &= e\left(g_0^{(1)}, \mathbf{h}\right),\end{aligned}$$

since $(g_0^{(1)} = g_1^{\pi_1(\mathrm{BU})\rho_1(\mathbf{s})}, \dots) \leftarrow \mathsf{SampG_1}(\mathrm{SP}_{\mathbb{G},1};\mathbf{s})$.

For the associative property we have to show that

$$\begin{aligned}e\left(g_0^{(1)}, h_i^{(1)}\right) &= e\left(g_1^{\pi_1(\mathrm{BU})\rho_1(\mathbf{s})}, g_2^{\pi_1(\mathrm{B}^*\mathrm{A}_i^\intercal)\rho_1(\mathbf{r})}\right)\\ &= e\left(g_1^{\pi_1(\mathrm{BA}_i\mathrm{U})\rho_1(\mathbf{s})}, g_2^{\pi_1(\mathrm{B}^*)\rho_1(\mathbf{r})}\right)\\ &= e\left(g_i^{(1)}, h_0^{(1)}\right).\end{aligned}$$

Observe that

$$(\mathrm{BU})^\intercal \mathrm{B}^* \mathrm{A}_i^\intercal = \mathrm{U}^\intercal \mathrm{B}^\intercal \mathrm{B}^* \mathrm{A}_i^\intercal = \mathrm{U}^\intercal \mathrm{A}_i^\intercal = (\mathrm{A}_i\mathrm{U})^\intercal \mathrm{B}^\intercal \mathrm{B}^* = (\mathrm{BA}_i\mathrm{U})^\intercal \mathrm{B}^*,$$

therefore we also have for the submatrices

$$(\pi_1(\mathrm{BU})\rho_1(\mathbf{s}))^\intercal \pi_1(\mathrm{B}^*\mathrm{A}_i^\intercal)\rho_1(\mathbf{r}) = (\pi_1(\mathrm{BA}_i\mathrm{U})\rho_1(\mathbf{s}))^\intercal \pi_1(\mathrm{B}^*)\rho_1(\mathbf{r}). \quad \square$$

Theorem 14. *The prime-order triple system groups construction satisfies the homomorphic property.*

Proof. Clearly,

$$\begin{aligned}\mathsf{SampG}_i(\mathrm{sp}_{\mathbb{G},i}, s_1) \cdot \mathsf{SampG}_i(\mathrm{sp}_{\mathbb{G},i}, s_2) \\ &= (g_1^{\pi_i(\mathrm{BU})\rho_i(\mathbf{s_1})}, \dots) \circ (g_1^{\pi_i(\mathrm{BU})\rho_i(\mathbf{s_2})}, \dots) \\ &= (g_1^{\pi_i(\mathrm{BU})(\rho_i(\mathbf{s_1})+\rho_i(\mathbf{s_2}))}, \dots) \\ &= \mathsf{SampG}_i(\mathrm{sp}_{\mathbb{G},i}, s_1 + s_2).\end{aligned}$$

□

Theorem 15. *The prime-order triple system groups construction satisfies the orthogonality property.*

Proof. We have to show that, for all $2 \leq i \leq k$,

$$\mu(\mathbf{h}^{(\mathbf{i})}) = \mu(g_2^{\pi_i(\mathrm{B}^*)\rho_i(\mathbf{r})}) = e\Big(g_1^{\pi_1(\mathrm{BU})}, g_2^{\pi_i(\mathrm{B}^*)\rho_i(\mathbf{r})}\Big) = \mathbb{1}.$$

Observe that

$$\pi_1(\mathrm{BU})^{\mathsf{T}}\pi_i(\mathrm{B}^*) = \rho_1(\mathrm{UB}^{\mathsf{T}})\pi_i(\mathrm{B}^*) = \mathbf{0},$$

for $2 \leq i \leq k$, and the proof follows directly. □

Theorem 16. *The prime-order triple system groups construction satisfies the non-degeneracy property.*

Proof. We have to show that $e\big(g_0^{(i)}, h_0^{(i)}\big)^a = e\big(g_1^{\pi_i(\mathrm{BU})\rho_i(\mathbf{s})}, g_2^{\pi_i(\mathrm{B}^*)\rho_i(\mathbf{r})}\big)^a \equiv g_T^b$. Observe that for $2 \leq i \leq k$

$$\begin{aligned}\big(\pi_i(\mathrm{BU})\rho_i(\mathbf{s})\big)^{\mathsf{T}}\pi_i(\mathrm{B}^*)\rho_i(\mathbf{r}) \cdot a &= \pi_i(\mathbf{s}^{\mathsf{T}})\rho_i(\mathrm{UB}^{\mathsf{T}})\pi_i(\mathrm{B}^*)\rho_i(\mathbf{r}) \cdot a \\ &= \pi_i(\mathbf{s}^{\mathsf{T}})\mathrm{I}_d\rho_i(\mathbf{r}) \cdot a \\ &\equiv b,\end{aligned}$$

and the proof follows directly. □

Lemma 8. *The prime-order triple system groups construction is $(\{1\}, \{1, c\})$-system group indistinguishable in $\mathbb{G}$ for $2 \leq c \leq k$, if no adversary can gain a non-negligible advantage in winning the generalized d-linear game (see Definition 15).*

Proof. We construct a challenger $\mathcal{B}$, capable of winning the generalized d-linear game with a non-negligible advantage, using an adversary $\mathcal{A}$, winning the $(\{1\}, \{1, c\})$-system group indistinguishability game in $\mathbb{G}$ with a non-negligible advantage.

In the proof description, we use values $\tilde{a}_{d+1}, \ldots, \tilde{a}_{2d}$ and $t_{d+1}, \ldots, t_{2d}$ which are initially left undefined. At the end of the proof, we explain how we can set these variables to obtain the desired theorem using a hybrid argument.

Setup Challenger $\mathcal{B}$ engages as adversary in the Setup of the generalized d-linear game to obtain the group parameters gp and the values $(g_1^{\mathbf{a}}, g_1^{a_{d+1}}, g_2^{\mathbf{a}})$. It sets $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T, e) = (\mathbb{G}_1^{3d}, \mathbb{G}_2^{3d}, \mathbb{G}_T, e)$ using the values from gp and indirectly sets U by picking $\tilde{u}_1, \ldots, \tilde{u}_d \xleftarrow{R} \mathbb{Z}_p^*$,

$$\mathrm{U} = \begin{pmatrix} a_1\tilde{u}_1 & & & & & \\ & \ddots & & & & \\ & & a_d\tilde{u}_d & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \quad \text{recall U has dimensions } 3d \times 3d.$$

The challenger sends $\mathrm{GP} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathbb{G}_r = (\mathbb{Z}_p)^{3d}, e, g_1, g_2, g_1^{\mathrm{U}})$ to the adversary and sets the counter value $q = 0$.

Query The challenger can answer queries for sampling parameters or for samples in $\mathbb{G}$ and $\mathbb{H}$. Before it does so, it first defines auxiliary matrices W and W^*. Let W be a sparse matrix with 1s across the diagonal and

$$\rho_c(\pi_1(\mathrm{W})) = \begin{pmatrix} a_1^{-1}\tilde{a}_{d+1} & \cdots & a_d^{-1}\tilde{a}_{d+1} \\ \vdots & \ddots & \vdots \\ a_1^{-1}\tilde{a}_{2d} & \cdots & a_d^{-1}\tilde{a}_{2d} \end{pmatrix}.$$

The matrix W^* is set to $(\mathrm{W}^{-1})^{\mathsf{T}}$.

For the sake of clarity, we show the matrices for the case $c = 2$,

$$\mathrm{W} = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ a_1^{-1}\tilde{a}_{d+1} & \cdots & a_d^{-1}\tilde{a}_{d+1} & 1 & & & & \\ \vdots & \ddots & \vdots & & \ddots & & & \\ a_1^{-1}\tilde{a}_{2d} & \cdots & a_d^{-1}\tilde{a}_{2d} & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix};$$

$$\mathrm{W}^* = (\mathrm{W}^{-1})^\intercal = \begin{pmatrix} 1 & & -a_1^{-1}\tilde{a}_{d+1} & \cdots & -a_1^{-1}\tilde{a}_{2d} & & & \\ & \ddots & \vdots & \ddots & \vdots & & & \\ & 1 & -a_d^{-1}\tilde{a}_{d+1} & \cdots & -a_d^{-1}\tilde{a}_{2d} & & & \\ & & 1 & & & & & \\ & & & \ddots & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix}.$$

Therefore, we have that

$$\mathrm{WU} = \begin{pmatrix} a_1\tilde{u}_1 & & & & & & & \\ & \ddots & & & & & & \\ & & a_d\tilde{u}_d & & & & & \\ \tilde{a}_{d+1}\tilde{u}_1 & \cdots & \tilde{a}_{d+1}\tilde{u}_d & 1 & & & & \\ \vdots & \ddots & \vdots & & \ddots & & & \\ \tilde{a}_{2d}\tilde{u}_1 & \cdots & \tilde{a}_{2d}\tilde{u}_d & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix}.$$

- The challenger answers the qth query for new sampling parameters upon receiving the parameters size n. The challenger sets

  $$\tilde{\mathrm{B}}_q \xleftarrow{R} \mathrm{GL}_{3d}(\mathbb{Z}_p), \tilde{\mathrm{B}}_q^* = (\tilde{\mathrm{B}}_q^{-1})^\intercal, \mathrm{B}_q = \tilde{\mathrm{B}}_q\mathrm{W}, \mathrm{B}_q^* = \tilde{\mathrm{B}}_q^*\mathrm{W}_q^*,$$
  $$\tilde{\mathrm{A}}_{q,0}, \ldots, \tilde{\mathrm{A}}_{q,n} \xleftarrow{R} (\mathbb{Z}_p)^{3d\times 3d}, \mathrm{A}_{q,i} = \mathrm{W}^{-1}\tilde{\mathrm{A}}_{q,i}\mathrm{W} \text{ for } i \in [n]^+.$$

  Note that matrices $\mathrm{B}_q, \mathrm{B}_q^*, \mathrm{A}_{q,0}, \ldots, \mathrm{A}_{q,n}$ are correctly distributed and that setting these variables implies

  $$\mathrm{B}_q\mathrm{A}_{q,i} = \tilde{\mathrm{B}}_q\mathrm{W}\mathrm{W}^{-1}\tilde{\mathrm{A}}_{q,i}\mathrm{W} = \tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,i}\mathrm{W}$$

  and

  $$\begin{aligned}\mathrm{B}_q^*\mathrm{A}_{q,i}^\intercal &= \tilde{\mathrm{B}}_q^*\mathrm{W}^*(\mathrm{W}^{-1}\tilde{\mathrm{A}}_{q,i}\mathrm{W})^\intercal \\ &= \tilde{\mathrm{B}}_q^*\mathrm{W}^*\mathrm{W}^\intercal\tilde{\mathrm{A}}_{q,i}^\intercal(\mathrm{W}^{-1})^\intercal \\ &= \tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal\mathrm{W}^*.\end{aligned}$$

  Now, the challenger $\mathcal{B}$ can set $\mathrm{SP}_{\mathbb{G},i}$ as

  $$g_1^{\pi_i(\mathrm{B}_q\mathrm{U})} = g_1^{\pi_i(\tilde{\mathrm{B}}_q\mathrm{W}\mathrm{U})} = g_1^{\tilde{\mathrm{B}}_q\pi_i(\mathrm{W}\mathrm{U})},$$
  $$g_1^{\pi_i(\mathrm{B}_q\mathrm{A}_{q,i}\mathrm{U})} = g_1^{\pi_i(\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,i}\mathrm{W}\mathrm{U})} = g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,i}\pi_i(\mathrm{W}\mathrm{U})},$$

using the values $\tilde{\mathrm{B}}_q$, $\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,i}$, and $g_1^{\pi_i(\mathrm{WU})}$.

The challenger $\mathcal{B}$ can also set $\mathrm{SP}_{\mathbb{H},i}$, for $i \neq c$, as

$$g_2^{\pi_i(\mathrm{B}_q^*)} = g_2^{\pi_i(\tilde{\mathrm{B}}_q^*\mathrm{W}^*)} = g_2^{\tilde{\mathrm{B}}_q^*\pi_i(\mathrm{W}^*)} = g_2^{\pi_i(\tilde{\mathrm{B}}_q^*)},$$
$$g_2^{\pi_i(\mathrm{B}_q^*\mathrm{A}_{q,i}^\intercal)} = g_2^{\pi_i(\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal\mathrm{W}^*)} = g_2^{\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal\pi_i(\mathrm{W}^*)} = g_2^{\pi_i(\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal)},$$

using the values $\tilde{\mathrm{B}}_q^*$ and $\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal$. Observe that we used the fact that $\pi_i(\mathrm{W}^*) = \pi_i(\mathrm{I}_{3d})$ for $i \neq c$.

Also note that $\mathcal{B}$ can set TR as

$$\pi_1(\mathrm{B}_q^*) = \pi_1(\tilde{\mathrm{B}}_q^*\mathrm{W}^*) = \tilde{\mathrm{B}}_q^*\pi_1(\mathrm{W}^*) = \pi_1(\tilde{\mathrm{B}}_q^*),$$
$$\pi_1(\mathrm{B}_q^*\mathrm{A}_{q,i}^\intercal) = \pi_1(\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal\mathrm{W}^*) = \tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal\pi_1(\mathrm{W}^*) = \pi_1(\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^\intercal).$$

Finally, the challenger $\mathcal{B}$ can define the map

$$\mu_q \colon \mathbf{h} \mapsto e\left(g_1^{\pi_1(\mathrm{B}_q\mathrm{U})}, \mathbf{h}\right) = e\left(g_1^{\tilde{\mathrm{B}}_q\pi_1(\mathrm{WU})}, \mathbf{h}\right).$$

- To answer a query for samples in $\mathbb{G}$ for the qth sampling parameters, the challenger $\mathcal{B}$ first queries in the generalized d-linear game for a challenge query. It receives the tuple

$$\left(g_1^{\mathbf{a}\circ\mathbf{t}}, g_1^{a_{d+1}\sum_{i=1}^d t_i + t_{d+1}}\right),$$

where either $t_{d+1} = 0$ or $t_{d+1} \in_R \mathbb{Z}_p^*$.

The challenger indirectly sets $\mathbf{s} \in (\mathbb{Z}_p)^{3d}$ by setting $s_1 = \tilde{r}_1^{-1}t_1, \ldots,$ $s_d = \tilde{r}_d^{-1}t_d$ and $s_{d(c-1)+1} = \tilde{t}_{d+1}, \ldots, s_{dc} = \tilde{t}_{dc}$, all other components are set to 0. Now, observe that $\tilde{\mathbf{s}} = \mathrm{WU}\mathbf{s}$ is a vector where $\tilde{s}_1 = a_1t_1, \ldots, \tilde{s}_d = a_dt_d$ and $\tilde{s}_{d(c-1)+1} = \tilde{a}_{d+1}\sum_{i=1}^d t_i + \tilde{t}_{d+1}, \ldots, \tilde{s}_{dc} = \tilde{a}_{2d}\sum_{i=1}^d t_i + \tilde{t}_{dc}$, all other components equal 0.

The challenger returns as challenge samples in $\mathbb{G}$ the values

$$\left(g_1^{\mathrm{B}_q\mathrm{U}\mathbf{s}}, g_1^{\mathrm{B}_q\mathrm{A}_{q,0}\mathrm{U}\mathbf{s}}, \ldots, g_1^{\mathrm{B}_q\mathrm{A}_{q,n}\mathrm{U}\mathbf{s}}\right) = \left(g_1^{\tilde{\mathrm{B}}_q\tilde{\mathbf{s}}}, g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,0}\tilde{\mathbf{s}}}, \ldots, g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,n}\tilde{\mathbf{s}}}\right).$$

Observe that if we have set $\tilde{a}_{d+1} = a_{d+1}$ and $\tilde{t}_{d+1} = t_{d+1}$, and set all other $\tilde{a}_i = 0$ and $\tilde{t}_i = 0$ for $d+1 < i \leq 2d$, we have constructed either a

sample $\mathbf{g}^{(\mathbf{1})}$ or some sample $\mathbf{g}^{(\mathbf{1})'}$ with 1 out of d elements from $\mathbf{g}^{(\mathbf{c})}$. Now, if we set $\tilde{a}_{d+1}, \tilde{t}_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$, set $\tilde{a}_{d+2} = a_{d+1}$ and $\tilde{t}_{d+2} = t_{d+1}$, and set all other $\tilde{a}_i = 0$ and $\tilde{t}_i = 0$ for $d+2 < i \leq 2d$, we have constructed either the sample $\mathbf{g}^{(\mathbf{1})'}$ or another sample with two elements from $\mathbf{g}^{(\mathbf{c})}$. So, following a hybrid argument using d hybrids, we can arrive at an indistinguishable game between samples $\mathbf{g}^{(\mathbf{1})}$ and sample $\mathbf{g}^{(\mathbf{1},\mathbf{c})}$. □

Theorem 17. *The prime-order triple system groups construction is* $(\{1\},\{1,2,3\})$*-system group indistinguishable in* $\mathbb{G}$ *if no adversary can gain a non-negligible advantage in winning the generalized* d*-linear game.*

Proof. From Lemma 8, we know that no p.p.t. adversary can gain a non-negligible advantage in winning the $(\{1\},\{1,c\})$-system group indistinguishable in $\mathbb{G}$ game (under the d-linear assumption).

Observe that the challenger can multiply the challenge $\mathbf{g}^{(\mathcal{L})}$ or $\mathbf{g}^{(\mathcal{R})}$ with $\mathbf{g}^{(\mathbf{j})}$ for any value $j \in [3] \backslash \{1, c\}$, since it knows $\text{SP}_{\mathbb{G},j}$. Using a hybrid argument, we thus obtain that $\mathbf{g}^{(\mathbf{1})} \approx_c \mathbf{g}^{(\mathbf{1},\mathbf{2})} \approx_c \mathbf{g}^{(\mathbf{1},\mathbf{2},\mathbf{3})}$. □

Theorem 18. *The prime-order triple system groups construction is* $(\mathcal{L},\mathcal{R},z)$*-system group indistinguishable in* $\mathbb{H}$ *if no adversary can gain a non-negligible advantage in winning the generalized* d*-linear game.*

Proof. We construct a challenger $\mathcal{B}$, capable of winning the generalized d-linear game with a non-negligible advantage, using an adversary $\mathcal{A}$, winning the $(\{z\},\{z,c\},z)$-system group indistinguishability game in $\mathbb{H}$ with a non-negligible advantage. Using a hybrid argument similar to the arugment used in Theorem 17, we obtain the desired theorem.

In the proof description, we use values $\tilde{a}_{d+1}, \ldots, \tilde{a}_{2d}$ and $t_{d+1}, \ldots, t_{2d}$ which are initially left undefined. At the end of the proof, we explain how we can set these variables to obtain the desired theorem using a hybrid argument.

Setup Challenger $\mathcal{B}$ engages as the adversary in the Setup of the generalized d-linear game to obtain the group parameters gp and values $(g_1^{\mathbf{a}}, g_2^{\mathbf{a}}, g_2^{a_{d+1}})$. It sets $(\mathbb{G},\mathbb{H},\mathbb{G}_T,e) = (\mathbb{G}_1^{3d},\mathbb{G}_2^{3d},\mathbb{G}_T,e)$ using the values from gp and indirectly sets U by picking $\tilde{u}_1, \ldots, \tilde{u}_d \xleftarrow{R} \mathbb{Z}_p^*$,

$$\mathrm{U} = \begin{pmatrix} a_1\tilde{u}_1 & & & & & & \\ & \ddots & & & & & \\ & & a_d\tilde{u}_d & & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 \end{pmatrix} \quad \text{recall U has dimensions } 3d \times 3d.$$

The challenger sends $\mathsf{GP} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \mathbb{G}_r = (\mathbb{Z}_p)^{3d}, e, g_1, g_2, g_1^{\mathrm{U}})$ to the adversary and sets the counter value $q = 0$.

Query The challenger can answer queries for sampling parameters or for samples in $\mathbb{G}$ and $\mathbb{H}$. Before it does so, it first defines auxiliary matrices W and W^*. Let W be a sparse matrix with 1s across the diagonal, except for

$$\rho_z(\pi_z(\mathrm{W})) = \begin{pmatrix} a_1^{-1} & & \\ & \ddots & \\ & & a_d^{-1} \end{pmatrix},$$

and

$$\rho_z(\pi_c(\mathrm{W})) = \begin{pmatrix} -a_1^{-1}\tilde{a}_{d+1} & \cdots & -a_1^{-1}\tilde{a}_{2d} \\ \vdots & \ddots & \vdots \\ -a_d^{-1}\tilde{a}_{d+1} & \cdots & -a_d^{-1}\tilde{a}_{2d} \end{pmatrix}.$$

The matrix W^* is set to $(\mathrm{W}^{-1})^{\mathsf{T}}$.

For the sake of clarity, we show the matrices for the case $c = 2$ and $z = 1$,

$$\mathrm{W} = \begin{pmatrix} a_1^{-1} & & & -a_1^{-1}\tilde{a}_{d+1} & \cdots & -a_1^{-1}\tilde{a}_{2d} & & & \\ & \ddots & & \vdots & \ddots & \vdots & & & \\ & & a_d^{-1} & -a_d^{-1}\tilde{a}_{d+1} & \cdots & -a_d^{-1}\tilde{a}_{2d} & & & \\ & & & 1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix};$$

$$\mathrm{W}^* = (\mathrm{W}^{-1})^{\mathsf{T}} = \begin{pmatrix} a_1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & a_d & & & & & & \\ \tilde{a}_{d+1} & \cdots & \tilde{a}_{d+1} & 1 & & & & & \\ \vdots & \ddots & \vdots & & \ddots & & & & \\ \tilde{a}_{2d} & \cdots & \tilde{a}_{2d} & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}.$$

Therefore, we have that

$$\mathrm{WU} = \begin{pmatrix} \tilde{u}_1 & & & -a_1^{-1}\tilde{a}_{d+1} & \cdots & -a_1^{-1}\tilde{a}_{2d} & & & \\ & \ddots & & \vdots & \ddots & \vdots & & & \\ & & \tilde{u}_d & -a_d^{-1}\tilde{a}_{d+1} & \cdots & -a_d^{-1}\tilde{a}_{2d} & & & \\ & & & 1 & & & & & \\ & & & & \ddots & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}.$$

- The challenger answers the qth query for new sampling parameters upon receiving the parameters size n. The challenger sets

$$\tilde{\mathrm{B}}_q \xleftarrow{R} \mathrm{GL}_{3d}(\mathbb{Z}_p), \tilde{\mathrm{B}}_q^* = (\tilde{\mathrm{B}}_q^{-1})^\intercal, \mathrm{B}_q = \tilde{\mathrm{B}}_q \mathrm{W}, \mathrm{B}_q^* = \tilde{\mathrm{B}}_q^* \mathrm{W}_q^*,$$
$$\tilde{\mathrm{A}}_{q,0}, \ldots, \tilde{\mathrm{A}}_{q,n} \xleftarrow{R} (\mathbb{Z}_p)^{3d\times 3d}, \mathrm{A}_{q,i} = \mathrm{W}^{-1}\tilde{\mathrm{A}}_{q,i}\mathrm{W} \text{ for } i \in [n]^+.$$

  Note that matrices $\mathrm{B}_q, \mathrm{B}_q^*, \mathrm{A}_{q,0}, \ldots, \mathrm{A}_{q,n}$ are correctly distributed and that setting these variables implies

$$\mathrm{B}_q \mathrm{A}_{q,i} = \tilde{\mathrm{B}}_q \mathrm{W}\mathrm{W}^{-1}\tilde{\mathrm{A}}_{q,i}\mathrm{W} = \tilde{\mathrm{B}}_q \tilde{\mathrm{A}}_{q,i}\mathrm{W}$$

  and

$$\begin{aligned} \mathrm{B}_q^* \mathrm{A}_{q,i}^\intercal &= \tilde{\mathrm{B}}_q^* \mathrm{W}^* (\mathrm{W}^{-1}\tilde{\mathrm{A}}_{q,i}\mathrm{W})^\intercal \\ &= \tilde{\mathrm{B}}_q^* \mathrm{W}^* \mathrm{W}^\intercal \tilde{\mathrm{A}}_{q,i}^\intercal (\mathrm{W}^{-1})^\intercal \\ &= \tilde{\mathrm{B}}_q^* \tilde{\mathrm{A}}_{q,i}^\intercal \mathrm{W}^*. \end{aligned}$$

  Now, the challenger $\mathcal{B}$ can set $\mathrm{sp}_{\mathbb{G},i}$, for $i \neq c$ and $i \neq z$ unless $z = 1$ (note $g_1^{\pi_1(\mathrm{WU})}$ can always be computed, even if $z = 1$), as

$$g_1^{\pi_i(\mathrm{B}_q \mathrm{U})} = g_1^{\pi_i(\tilde{\mathrm{B}}_q \mathrm{WU})} = g_1^{\tilde{\mathrm{B}}_q \pi_i(\mathrm{WU})},$$
$$g_1^{\pi_i(\mathrm{B}_q \mathrm{A}_{q,i} \mathrm{U})} = g_1^{\pi_i(\tilde{\mathrm{B}}_q \tilde{\mathrm{A}}_{q,i} \mathrm{WU})} = g_1^{\tilde{\mathrm{B}}_q \tilde{\mathrm{A}}_{q,i} \pi_i(\mathrm{WU})},$$

  using the values $\tilde{\mathrm{B}}_q$, $\tilde{\mathrm{B}}_q \tilde{\mathrm{A}}_{q,i}$, and $g_1^{\pi_i(\mathrm{WU})}$.

  The challenger $\mathcal{B}$ can also set $\mathrm{sp}_{\mathbb{H},i}$ as

$$g_2^{\pi_i(\mathrm{B}_q^*)} = g_2^{\pi_i(\tilde{\mathrm{B}}_q^* \mathrm{W}^*)} = g_2^{\tilde{\mathrm{B}}_q^* \pi_i(\mathrm{W}^*)},$$
$$g_2^{\pi_i(\mathrm{B}_q^* \mathrm{A}_{q,i}^\intercal)} = g_2^{\pi_i(\tilde{\mathrm{B}}_q^* \tilde{\mathrm{A}}_{q,i}^\intercal \mathrm{W}^*)} = g_2^{\tilde{\mathrm{B}}_q^* \tilde{\mathrm{A}}_{q,i}^\intercal \pi_i(\mathrm{W}^*)},$$

using the values $\tilde{\mathrm{B}}_q^*$, $\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,i}^{\intercal}$, and $g_2^{\pi_i(\mathrm{W}^*)}$.

Finally, the challenger $\mathcal{B}$ can define the map

$$\mu_q\colon \mathbf{h} \mapsto e\left(g_1^{\pi_1(\mathrm{B}_q\mathrm{U})}, \mathbf{h}\right) = e\left(g_1^{\tilde{\mathrm{B}}_q\pi_1(\mathrm{WU})}, \mathbf{h}\right).$$

- To answer a query for samples in $\mathbb{G}$ for the qth sampling parameters and value $j \in [k]$, the challenger $\mathcal{B}$ picks a vector $\tilde{\mathbf{v}} \in (\mathbb{Z}_p)^{3d}$ with $\tilde{v}_{d(i-1)+1}, \ldots, \tilde{v}_{di} \xleftarrow{R} \mathbb{Z}_p$ for $i \in \{z, j\}$ (note that we allow $z = j$) while all other vector components are set to 0. The vector $\tilde{\mathbf{v}}$ is used to indirectly set $\mathbf{s} = (\mathrm{WU})^{-1}\tilde{\mathbf{v}}$. Observe that $s_\eta \in_R \mathbb{Z}_p$ if and only if $\tilde{v}_\eta \in_R \mathbb{Z}_p$, and that the vector $\mathbf{s}$ is thus correctly distributed.

  The challenger sends

$$\begin{aligned}\mathbf{g}^{(\mathbf{z},\mathbf{j})} &= \left(g_1^{\mathrm{B}_q\mathrm{U}\mathbf{s}}, g_1^{\mathrm{B}_q\mathrm{A}_{q,0}\mathrm{U}\mathbf{s}}, \ldots, g_1^{\mathrm{B}_q\mathrm{A}_{q,n}\mathrm{U}\mathbf{s}}\right)\\ &= \left(g_1^{\tilde{\mathrm{B}}_q\mathrm{WU}(\mathrm{WU})^{-1}\tilde{\mathbf{v}}}, g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,0}\mathrm{WU}(\mathrm{WU})^{-1}\tilde{\mathbf{v}}}, \ldots,\right.\\ &\qquad\qquad \left. g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,n}\mathrm{WU}(\mathrm{WU})^{-1}\tilde{\mathbf{v}}}\right)\\ &= \left(g_1^{\tilde{\mathrm{B}}_q\tilde{\mathbf{v}}}, g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,0}\tilde{\mathbf{v}}}, \ldots, g_1^{\tilde{\mathrm{B}}_q\tilde{\mathrm{A}}_{q,n}\tilde{\mathbf{v}}}\right)\end{aligned}$$

  to the adversary.

- To answer a query for samples in $\mathbb{H}$ for the qth sampling parameters, the challenger $\mathcal{B}$ first queries in the generalized d-linear game for a challenge query. It receives the tuple

$$\left(g_2^{\mathbf{a}\circ\mathbf{t}}, g_2^{a_{d+1}\sum_{i=1}^{d} t_i + t_{d+1}}\right),$$

  where either $t_{d+1} = 0$ or $t_{d+1} \in_R \mathbb{Z}_p^*$.

  The challenger indirectly sets $\mathbf{r} \in (\mathbb{Z}_p)^{3d}$ by setting $r_{d(z-1)+1} = t_1, \ldots, r_{dz} = t_d$ and $r_{d(c-1)+1} = \tilde{t}_{d+1}, \ldots, r_{dc} = \tilde{t}_{dc}$, all other components are set to 0. Observe that $\tilde{\mathbf{r}} = \mathrm{W}^*\mathbf{r}$ is a vector where $\tilde{r}_{d(z-1)+1} = a_1 t_1, \ldots, \tilde{r}_{dz} = a_d t_d$ and $\tilde{r}_{d(c-1)+1} = \tilde{a}_{d+1}\sum_{i=1}^{d} t_i + \tilde{t}_{d+1}, \ldots, \tilde{r}_{dc} = \tilde{a}_{2d}\sum_{i=1}^{d} t_i + \tilde{t}_{dc}$, all other components equal 0.

  The challenger returns as challenge samples in $\mathbb{H}$ the values

$$\left(g_2^{\mathrm{B}_q^*\mathbf{r}}, g_2^{\mathrm{B}_q^*\mathrm{A}_{q,0}^{\intercal}\mathbf{r}}, \ldots, g_2^{\mathrm{B}_q^*\mathrm{A}_{q,n}^{\intercal}\mathbf{r}}\right) = \left(g_2^{\tilde{\mathrm{B}}_q^*\tilde{\mathbf{r}}}, g_2^{\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,0}^{\intercal}\tilde{\mathbf{r}}}, \ldots, g_2^{\tilde{\mathrm{B}}_q^*\tilde{\mathrm{A}}_{q,n}^{\intercal}\tilde{\mathbf{r}}}\right).$$

Observe that if we have set $\tilde{a}_{d+1} = a_{d+1}$ and $\tilde{t}_{d+1} = t_{d+1}$, and set all other $\tilde{a}_i = 0$ and $\tilde{t}_i = 0$ for $d+1 < i \leq 2d$, we have constructed either a sample $\mathbf{g}^{(\mathbf{z})}$ or some sample $\mathbf{g}^{(\mathbf{z})'}$ with one partial elements from $\mathbf{g}^{(\mathbf{c})}$. Now, if we set $\tilde{a}_{d+1}, \tilde{t}_{d+1} \xleftarrow{R} \mathbb{Z}_p^*$, set $\tilde{a}_{d+2} = a_{d+1}$ and $\tilde{t}_{d+2} = t_{d+1}$, and set all other $\tilde{a}_i = 0$ and $\tilde{t}_i = 0$ for $d+2 < i \leq 2d$, we have constructed either the sample $\mathbf{g}^{(\mathbf{z})'}$ or another sample with two elements from $\mathbf{g}^{(\mathbf{c})}$. So, following a hybrid argument using d hybrids, we can arrive at an indistinguishable game between samples $\mathbf{g}^{(\mathbf{z})}$ and sample $\mathbf{g}^{(\mathbf{z},\mathbf{c})}$. □

Theorem 19. *The prime-order triple system groups construction is parameter-hiding for all $i \geq 2$.*

Proof Intuition. To simplify the notation, we use here the vectors $\mathbf{s}$ and $\mathbf{r}$ of length d.

Note that

$$g_0^{(i)} = g_1^{\pi_i(\mathrm{B}_q\mathrm{U})\mathbf{s}}, g_j^{(i)} = g_1^{\pi_i(\mathrm{B}_q\mathrm{A}_{q,j}\mathrm{U})\mathbf{s}}$$
$$h_0^{(i)} = g_2^{\pi_i(\mathrm{B}_q^*)\mathbf{r}}, h_j^{(i)} = g_2^{\pi_i(\mathrm{B}_q^*\mathrm{A}_{q,j}^\intercal)\mathbf{r}}$$

and we can (uniformly) re-randomize $g_0^{(i)}$, $h_0^{(i)}$ respectively, using a vector $\boldsymbol{\gamma}_{q,j}$,

$$g_0^{(i)^{\gamma_{q,j}}} = g_1^{\pi_i(\mathrm{B}_q\mathrm{U})(\mathbf{s}\circ\boldsymbol{\gamma}_{q,j})} \qquad \text{and} \qquad h_0^{(i)^{\gamma_{q,j}}} = g_1^{\pi_i(\mathrm{B}_q^*)(\mathbf{r}\circ\boldsymbol{\gamma}_{q,j})}.$$

We have to show that

$$\left\{g_j^{(i)}, h_j^{(i)}\right\} \equiv \left\{g_j^{(i)} g_0^{(i)^{\gamma_{q,j}}}, h_j^{(i)} h_0^{(i)^{\gamma_{q,j}}}\right\}$$

for all j and q. Let $\tilde{\mathrm{A}}_{q,j}$ be a sparse matrix where we only set the submatrix $\rho_i(\pi_i(\tilde{\mathrm{A}}_{q,j}))$ of $\tilde{\mathrm{A}}_{q,j}$ to a random diagonal matrix in $\mathbb{Z}_p^{d\times d}$,

$$\rho_i(\pi_i(\tilde{\mathrm{A}}_{q,j})) = \mathrm{I}_d\boldsymbol{\gamma}_{q,j} = \begin{pmatrix} \gamma_{q,j,1} & & \\ & \ddots & \\ & & \gamma_{q,j,d} \end{pmatrix}.$$

Define $\mathrm{A}'_{q,j} = \mathrm{A}_{q,j} + \tilde{\mathrm{A}}_{q,j}$ and observe that

$$\begin{aligned} g_1^{\pi_i(\mathrm{B}_q\mathrm{A}'_{q,j}\mathrm{U})\mathbf{s}} &= g_1^{\pi_i(\mathrm{B}_q\mathrm{A}_{q,j}\mathrm{U}+\mathrm{B}_q\tilde{\mathrm{A}}_{q,j}\mathrm{U})\mathbf{s}} \\ &= g_1^{\pi_i(\mathrm{B}_q\mathrm{A}_{q,j}\mathrm{U})\mathbf{s}} g_1^{\pi_i(\mathrm{B}_q\tilde{\mathrm{A}}_{q,j}\mathrm{U})\mathbf{s}} && (\pi_i \text{ is linear}) \\ &= g_1^{\pi_i(\mathrm{B}_q\mathrm{A}_{q,j}\mathrm{U})\mathbf{s}} g_1^{\pi_i(\mathrm{B}_q\mathrm{U}\tilde{\mathrm{A}}_{q,j})\mathbf{s}} && (\tilde{\mathrm{A}}_{q,j} \text{ and } \mathrm{U} \text{ are diagonal}) \\ &= g_1^{\pi_i(\mathrm{B}_q\mathrm{A}_{q,j}\mathrm{U})\mathbf{s}} g_1^{\pi_i(\mathrm{B}_q\mathrm{U})\rho_i(\pi_i(\tilde{\mathrm{A}}_{q,j}))\mathbf{s}} && (\text{construction of } \tilde{\mathrm{A}}_{q,j}) \end{aligned}$$

$$= g_1^{\pi_i(\mathrm{B}_q \mathrm{A}_{q,j} \mathrm{U})\mathbf{s}} g_1^{\pi_i(\mathrm{B}_q \mathrm{U})(\boldsymbol{\gamma}_{q,j} \circ \mathbf{s})}$$

and

$$\begin{aligned} g_2^{\pi_i(\mathrm{B}_q^*(\mathrm{A}'_{q,j})^\intercal)\mathbf{r}} &= g_2^{\pi_i(\mathrm{B}_q^* \mathrm{A}_{q,j}^\intercal + \mathrm{B}_q^* \tilde{\mathrm{A}}_{q,j})\mathbf{r}} && (\tilde{\mathrm{A}}_{q,j} = \tilde{\mathrm{A}}_{q,j}^\intercal) \\ &= g_2^{\pi_i(\mathrm{B}_q^* \mathrm{A}_{q,j})\mathbf{r}} g_2^{\pi_i(\mathrm{B}_q^* \tilde{\mathrm{A}}_{q,j})\mathbf{r}} && (\pi_i \text{ is linear}) \\ &= g_2^{\pi_i(\mathrm{B}_q^* \mathrm{A}_{q,j})\mathbf{r}} g_2^{\pi_i(\mathrm{B}_q^*)\rho_i(\pi_i(\tilde{\mathrm{A}}_{q,j}))\mathbf{r}} && (\text{construction of } \tilde{\mathrm{A}}_{q,j}) \\ &= g_2^{\pi_i(\mathrm{B}_q^* \mathrm{A}_{q,j})\mathbf{r}} g_2^{\pi_i(\mathrm{B}_q^*)(\boldsymbol{\gamma}_{q,j} \circ \mathbf{r})}. \end{aligned}$$

Furthermore, observe that if $\boldsymbol{\gamma}_{q,j} = \mathbf{0}$ we obtain the first distribution and if $\boldsymbol{\gamma}_{q,j} \xleftarrow{R} \mathbb{Z}^d$ we obtain the second distribution. Finally, observe that we obtain identical distributions for $\mathrm{A}_{q,j}$ and $\mathrm{A}'_{q,j}$. □

6.2.8 Conversion from Encoding to Encryption

Now that we have defined TSG, we can give our conversion algorithm of a multi-authority admissible pair encoding scheme (MA-PES) to an MA-PE scheme in terms of the TSG.

We require that identities are unique random elements from the identity space $\mathcal{ID} = \mathbb{H}$. This may for example be achieve by choosing a cryptographic hash function $H\colon \{0,1\}^* \to \mathbb{H}$ and hash the ID to obtain a random element in $g_2^{\mathbf{r}} \in_R \mathbb{H}$.

GlobalSetup(1^λ)**.** The GlobalSetup algorithm first runs SampGroup(1^λ) to obtain the group parameters $(\mathrm{GP}, \mathrm{GS})$. The global public parameters pp are GP. The message space $\mathcal{M}$ is $\mathbb{G}_T$, the identity space $\mathcal{ID} = \mathbb{H}$.

AuthoritySetup$(\mathsf{pp}, \mathsf{par}_a)$**.** Given an MA-PES for par_a, the algorithm runs AuthorityParam(par_a) to obtain n. It uses SampP$(\mathrm{GP}, 1^n)$ to obtain

$$\left(\mu, \mathrm{SP}_{\mathbb{G},1}, \mathrm{SP}_{\mathbb{G},2}, \mathrm{SP}_{\mathbb{G},3}, \mathrm{SP}_{\mathbb{H},1}, \mathrm{SP}_{\mathbb{H},2}, \mathrm{SP}_{\mathbb{H},3}, \mathrm{TR}\right)$$

and picks $\mathsf{sk} \xleftarrow{R} \mathbb{H}$.

The authority's pk and msk are defined as

$$\mathsf{pk} = \left(\mathrm{SP}_{\mathbb{G},1}, \mu(\mathsf{sk})\right) \qquad \text{and} \qquad \mathsf{msk} = \left(\mathrm{TR}, \mathsf{sk}\right).$$

Encrypt$(\{(\mathsf{pk}_a, x_a)\}_{a \in \mathcal{A}}, m)$**.** Pick $\delta_a \xleftarrow{R} \mathbb{Z}_p$ for all $a \in \mathcal{A}$ and set the value $e(g_1, g_2)^\Delta = \prod_{a \in \mathcal{A}} e(g_1, g_2)^{\delta_a}$. Blind the message $m \in \mathbb{G}_T$ using $e(g_1, g_2)^\Delta$ to obtain $\mathsf{ct}_0 = m \cdot e(g_1, g_2)^\Delta$.

Pick an $a' \in \mathcal{A}$ and set $\omega_{a'} = -\sum_{a \in \mathcal{A} \setminus \{a'\}} \omega_a$, where $\omega_a \xleftarrow{R} \mathbb{G}_r$ for $a \neq a'$. Compute $(\Omega^{(1)}_{a,0}, \dots, \Omega^{(1)}_{a,n+1}) \leftarrow \mathsf{SampG_1}(\mathrm{SP}_{a,\mathbb{G},1}; \omega_a)$, for $a \in \mathcal{A}$, using the $\mathrm{SP}_{a,\mathbb{G},1}$ from pk_a.

Now, for each authority $a \in \mathcal{A}$ continue as follows (we frequently drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(p, x_a)$ to obtain w_1, w_2, and polynomials $(c_1, \dots, c_{w_3})$. Pick $s_0 \xleftarrow{R} \mathbb{G}_r$ and set $(g^{(1)}_{a,0,0}, \dots, g^{(1)}_{a,0,n+1}) \leftarrow \mathsf{SampG_1}(\mathrm{SP}_{a,\mathbb{G},1}; s_0)$. For $i \in [w_1 + w_2]$, sample $(g^{(1)}_{a,i,0}, \dots, g^{(1)}_{a,i,n+1}) \leftarrow \mathsf{SampG_1}(\mathrm{SP}_{a,\mathbb{G},1})$. Set $\mathsf{ct}_{a,1,i} = g^{(1)}_{a,i,0}$ for $i \in [w_1]^+$ and

$$\mathsf{ct}_{a,2,\ell} = \left(\Omega^{(1)}_{a,0}\right)^{\eta_\ell} \cdot \prod_{z \in [w_2]} \left(g^{(1)}_{a,w_1+z,0}\right)^{\eta_{\ell,z}} \cdot \prod_{i \in [w_1]^+, j \in [n]^+} \left(g^{(1)}_{a,i,j}\right)^{\eta_{\ell,i,j}}$$

for $\ell \in [w_3]$. Blind the value $e(g_1, g_2)^{\delta_a}$ using $\mathsf{SampGT}(\mu(\mathsf{sk}_a); s_0)$, by setting $\mathsf{ct}_{a,0} = e(g_1, g_2)^{\delta_a} \cdot \mathsf{SampGT}(\mu(\mathsf{sk}_a); s_0)$.

The complete ciphertext is

$$\mathsf{ct} = (\mathsf{ct}_0, \{\mathsf{ct}_{a,0}, \mathsf{ct}_{a,1,0}, \dots, \mathsf{ct}_{a,1,w_1}, \mathsf{ct}_{a,2,1}, \dots, \mathsf{ct}_{a,2,w_3}\}_{a \in \mathcal{A}}).$$

$\mathsf{KeyGen}(\mathsf{msk}_a, y, \mathsf{ID})$. The algorithm $\mathsf{EncKey}_a(p, y)$ is run to obtain m_1, m_2, and polynomials $(k_1, \dots, k_{m_3})$. Use the hash function H to compute $(h^{(1)}_{0,0}, \dots, h^{(1)}_{0,n+1}) \leftarrow \mathsf{MapH_1}(\mathrm{TR}, H(\mathsf{ID}))$ and sample the values $(h^{(1)}_{k,0}, \dots, h^{(1)}_{k,n+1}) \leftarrow \mathsf{SampH_1}(\mathrm{SP}_{a,\mathbb{H},1})$ for $k \in [m_1 + m_2]$. Set $\mathsf{usk}_{a,1,i} = h^{(1)}_{i,0}$ for $i \in [m_1]^+$ and

$$\mathsf{usk}_{a,2,\ell} = \mathsf{sk}_a^{\phi_\ell} \cdot \prod_{z \in [m_2]} \left(h^{(1)}_{m_1+z,0}\right)^{\phi_{\ell,z}} \cdot \prod_{i \in [m_1]^+, j \in [n]^+} \left(h^{(1)}_{i,j}\right)^{\phi_{\ell,i,j}}$$

for $\ell \in [m_3]$. The complete user secret key for $y \in \mathcal{Y}_{\kappa(a)}$ is

$$\mathsf{usk}_{y,\mathsf{ID}} = (\mathsf{usk}_{a,1,0}, \dots, \mathsf{usk}_{a,1,m_1}, \mathsf{usk}_{a,2,1}, \dots, \mathsf{usk}_{a,2,m_3}).$$

$\mathsf{Decrypt}(\{\mathsf{usk}_{y,\mathsf{ID}}\}_y, \mathsf{ct})$. To decrypt the ciphertext ct, we first decrypt $\mathsf{ct}_{a,0}$ for each authority $a \in \mathcal{A}$. Run $\mathsf{Pair}_a(p, x_a, y_a)$ to obtain E_a and $\hat{\mathrm{E}}_a$. Now compute

$$\mathsf{ct}_{a,0} \cdot \left(\prod_{\substack{i \in [w_1]^+, \\ \ell \in [m_3]}} e(\mathsf{ct}_{a,1,i}, \mathsf{usk}_{a,2,\ell})^{\mathrm{E}_{a,i,\ell}} \cdot \prod_{\substack{\ell \in [w_3], \\ i \in [m_1]^+}} e(\mathsf{ct}_{a,2,\ell}, \mathsf{usk}_{a,1,i})^{\hat{\mathrm{E}}_{a,\ell,\mathrm{i}}} \right)^{-1}$$

to obtain the result $e(g_1, g_2)^{\delta_a + \xi_a}$ for some value ξ_a. Note that these values ξ_a correspond to the values $\omega_a r_0$ of the pair encoding and that the values ω_a sum to 0. We can now combine these results to obtain

$$\prod_{a \in \mathcal{A}} e(g_1, g_2)^{\delta_a + \xi_a} = e(g_1, g_2)^{\Delta + 0} = e(g_1, g_2)^{\Delta},$$

and recover the plaintext $m = \mathsf{ct}_0 \cdot e(g_1, g_2)^{-\Delta}$.

6.2.9 Proving the Conversion Algorithm Secure

The proofs for the new conversion algorithm described above should be very similar to the proofs from Section 5.6. Sadly, we believe that it is not possible to simulate all values for corrupted authorities $a \in \bar{I}$ (*e.g.*, TR cannot always be simulated). However, if we omit the requirement to withstand corruptions, we believe the proofs can be delivered. For example, to prove Lemmas 2 and 3, we now do not use Assumption 4, but use indistinguishability in $\mathbb{G}$ (see Definition 17). For Lemmas 4 and 5 we use indistinguishability in $\mathbb{H}$ (Definition 18) and parameter-hiding (Definition 19). The final lemma, Lemma 6, can be proven using Assumption 8.

Observe that Lemma 5 in the original construction of Chapter 5 is based on Assumption 6, which corresponds in the new construction to requiring that an adversary cannot distinguish values $h^{(12)}$ from $h^{(13)}$. We claim this is the case. We can show that an adversary cannot distinguish values $h^{(1)}$ from $h^{(12)}$ (*i.e.*, $(\{1\},\{1,2\},2)$-System Group Indistinguishability in $\mathbb{H}$), cannot distinguish $h^{(12)}$ from $h^{(123)}$ ($(\{1,2\},\{1,2,3\},3)$-IND). Similarly, we have $(\{1\},\{1,3\},3)$-IND and $(\{1,3\},\{1,2,3\},2)$-IND. Hence, we can show using a hybrid argument that $h^{(12)}$ and $h^{(123)}$ are indistinguishable and that $h^{(13)}$ and $h^{(123)}$ are indistinguishable. So, we may conclude that $h^{(12)}$ and $h^{(13)}$ are indistinguishable.

We only show the proof for Lemma 6 here.

Lemma ($\mathsf{Game}_{2,q,2} \approx_c \mathsf{Game}_3$)**.** *Any p.p.t. adversary $\mathcal{A}$, making at most q key queries for distinct* IDs *and having at most a negligible advantage in breaking Assumption 8, has at most a negligible advantage in distinguishing $\mathsf{Game}_{2,q,2}$ from Game_3.*

Proof. The challenger $\mathcal{B}$ obtains $(g_1, g_2, g_2^x, g_1^y,\ g_1^{a_1}, \ldots, g_1^{a_d},\ g_2^{a_1}, \ldots, g_2^{a_d},\ g_1^{a_1 z_1}, \ldots, g_1^{a_d z_d},\ g_2^{a_1 z_1}, \ldots, g_2^{a_d z_d},\ T)$ where either $T = e(g_1, g_2)^{xy(z_1+\cdots+z_d)}$ or $T \in_R \mathbb{G}_T$. We write $\mathbf{a}$ for the column vector $(a_1, \ldots, a_d)^\mathsf{T}$, $\mathbf{x}$ for the column vector $(x, \ldots, x)^\mathsf{T}$ of length d, and similarly, $\mathbf{y}$ for $(y, \ldots, y)^\mathsf{T}$, and $\mathbf{z}$ for $(z_1, \ldots, z_d)^\mathsf{T}$.

We describe the phases of the security game.

Setup We run SampGroup as in the original scheme, setting g_1^{U} by choosing $\tilde{u}_1, \ldots, \tilde{u}_d \xleftarrow{R} \mathbb{Z}_p^*$ and setting $a_1\tilde{u}_1, \ldots, a_d\tilde{u}_d$ as the first d elements on the diagonal of U. Let $\bar{\mathrm{A}}$ be a sparse matrix with as only non-zero elements

$$\rho_3(\pi_3(\bar{\mathrm{A}})) = \mathrm{I}_d \cdot \mathbf{z} = \begin{pmatrix} z_1 & & \\ & \ddots & \\ & & z_d \end{pmatrix}.$$

Hash Oracle Upon receiving oracle query ID for the hash function H, the challenger $\mathcal{B}$ checks if it received the query before, and if so, answers with the same reply as before. If $\mathcal{A}$ has not queried for the hash value of ID before, $\mathcal{B}$ picks vectors $\mathbf{r}_{0,\mathsf{ID},1}, \tilde{\mathbf{r}}_{0,\mathsf{ID},3} \xleftarrow{R} (\mathbb{Z}_p)^d$ and sets $\mathbf{r}_{0,\mathsf{ID},3} = -\mathbf{x} + \mathbf{a} \circ \tilde{\mathbf{r}}_{0,\mathsf{ID},3}$. It answers the query with $g_2^{\mathbf{r}_{0,\mathsf{ID},1}}$.

Authority Queries Request for a new authority a using par_a are answered by the challenger by first running $\mathsf{AuthorityParam}(\mathsf{par}_a)$ to obtain n. Next, the challenger (indirectly) sets

$$\mathrm{B}_a \xleftarrow{R} \mathrm{GL}_{3d}(\mathbb{Z}_p),\ \mathrm{B}_a^* = (\mathrm{B}_a^{-1})^{\mathsf{T}},\ \tilde{\mathrm{A}}_{a,0}, \mathrm{A}_{a,1}, \dots, \mathrm{A}_{a,n} \xleftarrow{R} (\mathbb{Z}_p)^{3d \times 3d},$$
$$\mathrm{A}_{a,0} = \tilde{\mathrm{A}}_{a,0} + \bar{\mathrm{A}}.$$

The challenger picks $\boldsymbol{\alpha}_{a,1}, \boldsymbol{\alpha}_{a,2}, \tilde{\boldsymbol{\alpha}}_{a,3} \xleftarrow{R} (\mathbb{Z}_p)^d$ and indirectly sets $\boldsymbol{\alpha}_{a,3} = (x\mathbf{z} - \tilde{\boldsymbol{\alpha}}_{a,3})$. Let $\boldsymbol{\alpha}_a$ be the vector $(\boldsymbol{\alpha}_{a,1}; \boldsymbol{\alpha}_{a,2}; \boldsymbol{\alpha}_{a,3})$. The challenger gives $\mathcal{A}$ the public key $\mathsf{pk} = (\mathrm{sp}_{\mathbb{G},1}, \mu(\mathsf{sk}))$ as

$$\begin{aligned}
&\Big(g_1^{\pi_1(\mathrm{B}_a\mathrm{U})}, g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,0}\mathrm{U})}, \dots, g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,n}\mathrm{U})}, e\Big(g_1^{\pi_1(\mathrm{B}_a\mathrm{U})}, g_2^{\mathrm{B}_a^*\boldsymbol{\alpha}_a}\Big)\Big) \\
&= \Big(g_1^{\pi_1(\mathrm{B}_a\mathrm{U})}, g_1^{\pi_1(\mathrm{B}_a(\tilde{\mathrm{A}}_{a,0}+\bar{\mathrm{A}})\mathrm{U})}, g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,1}\mathrm{U})}, \dots, g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,n}\mathrm{U})}, \\
&\qquad\qquad e\Big(g_1^{\pi_1(\mathrm{B}_a\mathrm{U})}, g_2^{\pi_1(\mathrm{B}_a^*)\rho_1(\boldsymbol{\alpha}_a)}\Big)\Big) \\
&= \Big(g_1^{\pi_1(\mathrm{B}_a\mathrm{U})}, g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,0}\mathrm{U})} \cdot g_1^{\pi_1(\mathrm{B}_a\bar{\mathrm{A}}\mathrm{U})}, g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,1}\mathrm{U})}, \dots, \\
&\qquad\qquad g_1^{\pi_1(\mathrm{B}_a\mathrm{A}_{a,n}\mathrm{U})}, e\Big(g_1^{\rho_1(\pi_1(\mathrm{U}))}, g_2^{\boldsymbol{\alpha}_{a,1}}\Big)\Big) \\
&= \Big(g_1^{\mathrm{B}_a\pi_1(\mathrm{U})}, g_1^{\mathrm{B}_a\mathrm{A}_{a,0}\pi_1(\mathrm{U})}, g_1^{\mathrm{B}_a\mathrm{A}_{a,1}\pi_1(\mathrm{U})}, \dots, g_1^{\mathrm{B}_a\mathrm{A}_{a,n}\pi_1(\mathrm{U})}, \\
&\qquad\qquad e\Big(g_1^{\rho_1(\pi_1(\mathrm{U}))}, g_2^{\boldsymbol{\alpha}_{a,1}}\Big)\Big)
\end{aligned}$$

and adds a to the set I. Note that $g_1^{\pi_1(\mathrm{U})}$ can be computed using $g_1^{\mathbf{a}}$.

Key Queries Upon receiving a key query $(a, y \in \mathcal{Y}_{\kappa(a)}, \mathsf{ID})$ for an uncorrupted authority $a \in I$, $\mathcal{B}$ answers the query with a semi-functional key of type II. The challenger $\mathcal{B}$ computes $\overline{\mathsf{KeyGen}}(\mathsf{sk}_a, y; (1,3), \mathbf{u}_{\mathsf{ID}})$ as follows. First, it sets

$$\begin{aligned}
\mathsf{usk}_{a,1,0} &= g_2^{\pi_1(\mathrm{B}_a^*)\mathbf{r}_{0,\mathsf{ID},1} + \pi_3(\mathrm{B}_a^*)\mathbf{r}_{0,\mathsf{ID},3}} \\
&= g_2^{\pi_1(\mathrm{B}_a^*)\mathbf{r}_{0,\mathsf{ID},1} + \pi_3(\tilde{\mathrm{B}}_a^*)(-\mathbf{x} + \mathbf{a} \circ \tilde{\mathbf{r}}_{0,\mathsf{ID},3})},
\end{aligned}$$

and for $i \in [m_1 + m_2]$, $\mathsf{usk}_{a,1,i} = g_2^{\pi_1(\mathrm{B}_a^*)\mathbf{r}_{i,1}+\pi_3(\mathrm{B}_a^*)\mathbf{r}_{i,3}}$, where $\mathbf{r}_{i,1}, \tilde{\mathbf{r}}_{i,3} \xleftarrow{R} (\mathbb{Z}_p)^d$ and $\mathbf{r}_{i,3} = \mathbf{a} \circ \tilde{\mathbf{r}}_{i,3}$.

Next, to construct the values $\mathsf{usk}_{a,2,\ell}$, consider two cases. Either the encoding k_ℓ contains both the symbols α and $b_0 r_0$, or it does not contain this combination (*i.e.*, $\phi_\ell = \phi_{\ell,0,0}$, see Section 5.4.1).

In the case that α and $b_0 r_0$ do not occur in k_ℓ, $\mathcal{B}$ can readily create $\mathsf{usk}_{a,2,\ell}$ using the values $\mathsf{usk}_{a,1,m_1+1}, \ldots, \mathsf{usk}_{a,1,m_1+m_2}$ (corresponding to $\hat{r}_i$), values (corresponding to $b_0 r_i$ for $i \in [m_1]$)

$$\begin{aligned}
&g_2^{\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,1}^\intercal)\mathbf{r}_{i,1}+\pi_3(\mathrm{B}_a^*\mathrm{A}_{a,1}^\intercal)\mathbf{r}_{i,3}} \\
&= g_2^{\pi_1(\mathrm{B}_a^*(\tilde{\mathrm{A}}_{a,1}+\bar{\mathrm{A}})^\intercal)\mathbf{r}_{i,1}+\pi_3(\mathrm{B}_a^*(\tilde{\mathrm{A}}_{a,1}+\bar{\mathrm{A}})^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{i,3})} \\
&= g_2^{\pi_1(\mathrm{B}_a^*\tilde{\mathrm{A}}_{a,1}^\intercal)\mathbf{r}_{i,1}} g_2^{\pi_3(\mathrm{B}_a^*\tilde{\mathrm{A}}_{a,1}^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{i,3})} g_2^{\pi_3(\mathrm{B}_a^*\bar{\mathrm{A}}^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{i,3})},
\end{aligned}$$

where $g_2^{\pi_3(\mathrm{B}^*\tilde{\mathrm{A}}_{a,1}^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{i,3})}$ can be constructed using $g_2^{\mathbf{a}}$ and $g_2^{\pi_3(\mathrm{B}^*\bar{\mathrm{A}}^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{i,3})}$ using $g_2^{\mathbf{a}\circ\mathbf{z}}$, the values (corresponding to $b_j r_0$ for $j \in [n]$)

$$\begin{aligned}
&g_2^{\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)\mathbf{r}_{0,\mathsf{ID},1}+\pi_3(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)\mathbf{r}_{0,\mathsf{ID},3}} \\
&\qquad\qquad = g_2^{\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)\mathbf{r}_{0,\mathsf{ID},1}+\pi_3(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)(-\mathbf{x}+\mathbf{a}\circ\tilde{\mathbf{r}}_{0,\mathsf{ID},3})},
\end{aligned}$$

which can be constructed using g_2^x and $g_2^{\mathbf{a}}$, and the values (corresponding to $b_j r_i$ for $i \in [m_1]$ and $j \in [n]$)

$$g_2^{\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)\mathbf{r}_{i,1}+\pi_3(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)\mathbf{r}_{i,3}} = g_2^{\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,j}^\intercal)\mathbf{r}_{i,1}+\pi_3(\tilde{\mathrm{B}}_a^*\mathrm{A}_{a,j}^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{i,3})},$$

which can be computed using $g_2^{\mathbf{a}}$.

In the case that both α and $b_0 r_0$ occur in k_ℓ, observe that $\mathcal{B}$ needs to compute

$$\begin{aligned}
&g_2^{\phi_\ell\left(\mathrm{B}_a^*\boldsymbol{\alpha}_a\right)+\phi_{\ell,0,0}\left(\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,0}^\intercal)\mathbf{r}_{0,\mathsf{ID},1}+\pi_3(\mathrm{B}_a^*\mathrm{A}_{a,0}^\intercal)\mathbf{r}_{0,\mathsf{ID},3}\right)} \\
&= g_2^{\phi_\ell\left(\pi_1(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,1}+\pi_2(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,2}+\pi_3(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,3}+\pi_1(\mathrm{B}_a^*\mathrm{A}_{a,0}^\intercal)\mathbf{r}_{0,\mathsf{ID},1}+\pi_3(\mathrm{B}_a^*\mathrm{A}_{a,0}^\intercal)\mathbf{r}_{0,\mathsf{ID},3}\right)} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\text{since } \phi_\ell = \phi_{\ell,0,0}) \\
&= g_2^{\phi_\ell\left(\pi_1(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,1}+\pi_2(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,2}+\pi_3(\mathrm{B}_a^*)(x\mathbf{z}-\tilde{\boldsymbol{\alpha}}_{a,3})\right)} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot g_2^{\phi_\ell\mathrm{B}_a^*\left(\pi_1(\tilde{\mathrm{A}}_{a,0}^\intercal+\bar{\mathrm{A}}^\intercal)\mathbf{r}_{0,\mathsf{ID},1}+\pi_3(\tilde{\mathrm{A}}_{a,0}^\intercal+\bar{\mathrm{A}}^\intercal)(-\mathbf{x}+\mathbf{a}\circ\tilde{\mathbf{r}}_{0,\mathsf{ID},3})\right)} \\
&= g_2^{\phi_\ell\left(\pi_1(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,1}+\pi_2(\mathrm{B}_a^*)\boldsymbol{\alpha}_{a,2}-\pi_3(\mathrm{B}_a^*)\tilde{\boldsymbol{\alpha}}_{a,3}\right)} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot g_2^{\phi_\ell\mathrm{B}_a^*\left(\pi_1(\tilde{\mathrm{A}}_{a,0}^\intercal)\mathbf{r}_{0,\mathsf{ID},1}+\pi_3(\tilde{\mathrm{A}}_{a,0}^\intercal)(-\mathbf{x}+\mathbf{a}\circ\tilde{\mathbf{r}}_{0,\mathsf{ID},3})+\pi_3(\bar{\mathrm{A}}^\intercal)(\mathbf{a}\circ\tilde{\mathbf{r}}_{0,\mathsf{ID},3})\right)},
\end{aligned}$$

where in the last step we have used the fact that $\pi_3(\mathrm{B}_a^*)\cdot x\mathbf{z} - \pi_3(\mathrm{B}_a^*\bar{\mathrm{A}})\mathbf{x} = \mathbf{0}$. Note that in this case, we can compute $\mathsf{usk}_{a,2,\ell}$ too, using g_2^x, $g_2^{\mathbf{a}}$, and $g_2^{\mathbf{a}\circ\mathbf{z}}$.

Furthermore, note that the key queries are created with properly distributed semi-functional keys of type II.

Challenge Ciphertext Whenever $\mathcal{A}$ requests the ciphertext challenge by sending $(m_0, m_1, \{x_a^*\}_{a\in\mathcal{A}^*})$, the challenger $\mathcal{B}$ picks $b \xleftarrow{R} \{0,1\}$ and encrypts message m_b as a semi-functional challenge ciphertext.

Choose an authority $a' \in \mathcal{A}^*$. For each authority $a \in \mathcal{A}^* \setminus a'$, pick $\boldsymbol{\omega}_{a,1} \xleftarrow{R} (\mathbb{Z}_p)^d$ and $\delta_a \xleftarrow{R} \mathbb{Z}_p$, and set $\boldsymbol{\omega}_{a',1} = -\sum_{a\in\mathcal{A}^*\setminus a'} \boldsymbol{\omega}_{a,1}$ and indirectly set $\delta_{a'} = xy(z_1+\cdots+z_d) - \sum_{a\in\mathcal{A}^*\setminus a'} \delta_a$. Additionally, pick $\boldsymbol{\omega}_{a,2}, \tilde{\boldsymbol{\omega}}_{a,3} \xleftarrow{R} (\mathbb{Z}_p)^d$ for all $a \in \mathcal{A}^*$ and (indirectly) set $\boldsymbol{\omega}_{a'} = (\boldsymbol{\omega}_{a',1}; \boldsymbol{\omega}_{a',2}; y\mathbf{z} + \tilde{\boldsymbol{\omega}}_{a',3})$ and $\boldsymbol{\omega}_a = (\boldsymbol{\omega}_{a,1}; \boldsymbol{\omega}_{a,2}; \tilde{\boldsymbol{\omega}}_{a,3})$ for all $a \in \mathcal{A}^* \setminus a'$. Blind the message $m_b \in \mathbb{G}_T$ using T to obtain $\mathsf{ct}_0 = m_b \cdot T$. Note that if $T = e(g_1,g_2)^{xy(z_1+\cdots+z_d)}$, the challenger simulates $\mathsf{Game}_{2,q,2}$ using $\Delta = xy(z_1+\cdots+z_d)$ and otherwise, if $T \in_R \mathbb{G}_T$, the challenger simulates Game_3.

Now, for each authority $a \in \mathcal{A}^*$ continue as follows (we frequently drop the index a—when there is no ambiguity—to simplify notation). Run $\mathsf{EncCt}_a(p,x)$ to obtain w_1, w_2, and polynomials $(c_1,\ldots,c_{w_3})$.

If $a = a'$, pick $\mathbf{s}_{a',i,1}, \mathbf{s}_{a',i,2}, \tilde{\mathbf{s}}_{a',i,3} \xleftarrow{R} (\mathbb{Z}_p)^d$ for $i \in [w_1+w_2]^+$. Set $\mathbf{s}_{a',0,3} = -\mathbf{y} + \mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}$ and for $i \in [w_1+w_2]$, set $\mathbf{s}_{a',i,3} = \mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3}$. Set

$$\begin{aligned}\mathsf{ct}_{a',1,0} &= g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',0,1} + \pi_2(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',0,2} + \pi_3(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',0,3}} \\ &= g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',0,1} + \pi_2(\mathrm{B}_{a'})\mathbf{s}_{a',0,2} + \pi_3(\mathrm{B}_{a'})(-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3})}.\end{aligned}$$

Observe that $\mathsf{ct}_{a',1,0}$ can be computed using g_1^y and $g_1^{\mathbf{a}}$. The elements

$$\begin{aligned}\mathsf{ct}_{a',1,i} &= g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',i,1} + \pi_2(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',i,2} + \pi_3(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',i,3}} \\ &= g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',i,1} + \pi_2(\mathrm{B}_{a'})\mathbf{s}_{a',i,2} + \pi_3(\mathrm{B}_{a'})(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3})}.\end{aligned}$$

for $i \in [w_1]$ can also readily be computed using $g_1^{\mathbf{a}}$.

Next, to construct the values $\mathsf{ct}_{a',2,\ell}$, consider two cases. Either the encoding c_ℓ contains both the symbol ω and $b_0 s_0$, or it does not contain this combination (*i.e.*, $\eta_\ell = \eta_{\ell,0,0}$, see Section 5.4.1).

In the case that ω and $b_0 s_0$ do not occur in c_ℓ, $\mathcal{B}$ can readily create

$$g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{U})\mathbf{s}_{a',i,1} + \pi_2(\mathrm{B}_{a'})\mathbf{s}_{a',i,2} + \pi_3(\mathrm{B}_{a'})(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3})}$$

for $w_1+1 \le i \le w_1+w_2$ (corresponding to $\hat{s}_i$), the values

$$g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{A}_{a',0}\mathrm{U})\mathbf{s}_{a',i,1} + \pi_2(\mathrm{B}_{a'}\mathrm{A}_{a',0}\mathrm{U})\mathbf{s}_{a',i,2} + \pi_3(\mathrm{B}_{a'}\mathrm{A}_{a',0}\mathrm{U})\mathbf{s}_{a',i,3}}$$

$$= g_1^{\mathrm{B}_{a'}\left(\pi_1((\tilde{\mathrm{A}}_{a',0}+\bar{\mathrm{A}})\mathrm{U})\mathbf{s}_{a',i,1}+\pi_2((\tilde{\mathrm{A}}_{a',0}+\bar{\mathrm{A}}))\mathbf{s}_{a',i,2}+\pi_3((\tilde{\mathrm{A}}_{a',0}+\bar{\mathrm{A}}))(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3})\right)}$$
$$= g_1^{\mathrm{B}_{a'}\left(\pi_1(\tilde{\mathrm{A}}_{a',0}\mathrm{U})\mathbf{s}_{a',i,1}+\pi_2(\tilde{\mathrm{A}}_{a',0})\mathbf{s}_{a',i,2}+\pi_3(\tilde{\mathrm{A}}_{a',0})(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3})+\pi_3(\bar{\mathrm{A}})(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3})\right)},$$

for $i \in [w_1]$ (corresponding to $b_0 s_i$), the values

$$g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',0,1}+\pi_2(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',0,2}+\pi_3(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',0,3}}$$
$$= g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',0,1}+\pi_2(\mathrm{B}_{a'}\mathrm{A}_{a',j})\mathbf{s}_{a',0,2}+\pi_3(\mathrm{B}_{a'}\mathrm{A}_{a',j})(-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3})},$$

for $j \in [n]$ (corresponding to $b_j s_0$), and the values

$$g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',i,1}+\pi_2(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',i,2}+\pi_3(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',i,3}}$$
$$= g_1^{\pi_1(\mathrm{B}_{a'}\mathrm{A}_{a',j}\mathrm{U})\mathbf{s}_{a',i,1}+\pi_2(\mathrm{B}_{a'}\mathrm{A}_{a',j})\mathbf{s}_{a',i,2}+\pi_3(\mathrm{B}_{a'}\mathrm{A}_{a',j})(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',i,3})},$$

for $i \in [w_1]$ and $j \in [n]$ (corresponding to $b_j s_i$).

In the case that both ω and $b_0 s_0$ occur in c_ℓ, observe that $\mathcal{B}$ needs to compute

$$\begin{aligned}
&g_1^{\eta_\ell(\mathrm{B}_{a'}\mathrm{U}\boldsymbol{\omega}_{a'})+\eta_{\ell,0,0}\left(\mathrm{B}_{a'}\mathrm{A}_{a',0}\mathrm{U}\left(\mathbf{s}_{a',0,1};\mathbf{s}_{a',0,2};\mathbf{s}_{a',0,3}\right)\right)}\\
&= g_1^{\eta_\ell\mathrm{B}_{a'}\left(\mathrm{U}\boldsymbol{\omega}'_{a'}+\mathrm{A}_{a',0}\mathrm{U}\left(\mathbf{s}_{a',0,1};\mathbf{s}_{a',0,2};-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}\right)\right)} \qquad (\text{since } \eta_\ell = \eta_{\ell,0,0})\\
&= g_1^{\eta_\ell\mathrm{B}_{a'}\left(\mathrm{U}\left(\boldsymbol{\omega}_{a',1};\boldsymbol{\omega}_{a',2};y\mathbf{z}+\tilde{\boldsymbol{\omega}}_{a',3}\right)+(\tilde{\mathrm{A}}_{a',0}+\bar{\mathrm{A}})\mathrm{U}\left(\mathbf{s}_{a',0,1};\mathbf{s}_{a',0,2};-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}\right)\right)}\\
&= g_1^{\eta_\ell\mathrm{B}_{a'}\left(\mathrm{U}\left(\boldsymbol{\omega}_{a',1};\boldsymbol{\omega}_{a',2};\tilde{\boldsymbol{\omega}}_{a',3}\right)+\tilde{\mathrm{A}}_{a',0}\mathrm{U}\left(\mathbf{s}_{a',0,1};\mathbf{s}_{a',0,2};-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}\right)+\pi_3(\bar{\mathrm{A}})(\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3})\right)}
\end{aligned}$$

where in the last step we have used the fact that $\pi_3(\mathrm{U})\cdot y\mathbf{z} - \pi_3(\bar{\mathrm{A}}\mathrm{U})\mathbf{y} = \mathbf{0}$. Note that in this case, we can compute $\mathsf{ct}_{a',2,\ell}$ too, using the values g_1^y, $g_1^{\mathbf{a}}$, and $g_1^{\mathbf{a}\circ\mathbf{z}}$.

The challenger blinds the value $e(g_1,g_2)^{\delta_{a'}}$ by setting $\mathsf{ct}_{a',0}$ as

$$\begin{aligned}
&e(g_1,g_2)^{\delta_{a'}}\cdot e\left(g_1^{\mathrm{B}_{a'}\mathrm{U}\left(\mathbf{s}_{a',0,1};\mathbf{s}_{a',0,2};\mathbf{s}_{a',0,3}\right)}, g_2^{\mathrm{B}^*_{a'}\boldsymbol{\alpha}_{a'}}\right)\\
&= e(g_1,g_2)^{xy(z_1+\cdots+z_d)-\sum_{a\in\mathcal{A}^*\setminus a'}\delta_a}\\
&\qquad\cdot e\left(g_1^{\mathrm{B}_{a'}\mathrm{U}\left(\mathbf{s}_{a',0,1};\mathbf{s}_{a',0,2};-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}\right)}, g_2^{\mathrm{B}^*_{a'}\left(\boldsymbol{\alpha}_{a',1};\boldsymbol{\alpha}_{a',2};x\mathbf{z}-\tilde{\boldsymbol{\alpha}}_{a',3}\right)}\right)\\
&= e(g_1,g_2)^{xy(z_1+\cdots+z_d)-\sum_{a\in\mathcal{A}^*\setminus a'}\delta_a}\\
&\qquad\cdot e\left(g_1^{\rho_1(\pi_1(\mathrm{U}))\mathbf{s}_{a',0,1}}, g_2^{\boldsymbol{\alpha}_{a',1}}\right)\cdot e\left(g_1^{\rho_2(\pi_2(\mathrm{U}))\mathbf{s}_{a',0,2}}, g_2^{\boldsymbol{\alpha}_{a',2}}\right)\\
&\qquad\qquad\cdot e\left(g_1^{\rho_3(\pi_3(\mathrm{U}))(-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3})}, g_2^{x\mathbf{z}-\tilde{\boldsymbol{\alpha}}_{a',3}}\right)
\end{aligned}$$

$$= e(g_1, g_2)^{-\sum_{a \in \mathcal{A}^* \setminus a'} \delta_a} \cdot e\Big(g_1^{\rho_1(\pi_1(\mathrm{U}))\mathbf{s}_{a',0,1}}, g_2^{\boldsymbol{\alpha}_{a',1}}\Big) \cdot e\Big(g_1^{\mathbf{s}_{a',0,2}}, g_2^{\boldsymbol{\alpha}_{a',2}}\Big) \cdot e\Big(g_1^{\mathbf{a} \circ \tilde{\mathbf{s}}_{a',0,3}}, g_2^{x\mathbf{z}}\Big) \cdot e\Big(g_1^{-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}}, g_2^{\tilde{\boldsymbol{\alpha}}_{a',3}}\Big)$$

$$= e(g_1, g_2)^{-\sum_{a \in \mathcal{A}^* \setminus a'} \delta_a} \cdot e\Big(g_1^{\rho_1(\pi_1(\mathrm{U}))\mathbf{s}_{a',0,1}}, g_2^{\boldsymbol{\alpha}_{a',1}}\Big) \cdot e\Big(g_1^{\mathbf{s}_{a',0,2}}, g_2^{\boldsymbol{\alpha}_{a',2}}\Big) \cdot e\Big(g_1^{\mathbf{a}\circ\mathbf{z}\circ\tilde{\mathbf{s}}_{a',0,3}}, g_2^{\mathbf{x}}\Big) \cdot e\Big(g_1^{-\mathbf{y}+\mathbf{a}\circ\tilde{\mathbf{s}}_{a',0,3}}, g_2^{\tilde{\boldsymbol{\alpha}}_{a',3}}\Big),$$

which can be computed using the values $g_1^{\mathbf{a}}$, $g_1^{\mathbf{a}\circ\mathbf{z}}$, g_2^x, and g_1^y.

If $a \neq a'$, pick $\mathbf{s}_{a,i,1}, \mathbf{s}_{a,i,2}, \tilde{\mathbf{s}}_{a,i,3} \xleftarrow{R} (\mathbb{Z}_p)^d$ for $i \in [w_1 + w_2]^+$ and set $\mathbf{s}_{a,i} = (\mathbf{s}_{a,i,1}; \mathbf{s}_{a,i,2}; \mathbf{a} \circ \tilde{\mathbf{s}}_{a,i,3})$. Set $\mathsf{ct}_{a,1,i} = g_1^{\mathrm{B}_a \mathrm{U} \mathbf{s}_{a,i}}$, for $i \in [w_1]^+$, which can be computed using $g_1^{\mathbf{a}}$.

To construct the values $\mathsf{ct}_{a,2,\ell}$, we use $g_1^{\mathrm{B}_a \mathrm{U} \mathbf{s}_{a,i}}$ for $w_1 + 1 \leq i \leq w_1 + w_2$ (corresponding to $\hat{s}_i$), which can be computed using $g_1^{\mathbf{a}}$. We use

$$g_1^{\mathrm{B}_a \mathrm{A}_{a,0} \mathrm{U} \mathbf{s}_{a,i}} = g_1^{\mathrm{B}_a (\tilde{\mathrm{A}}_{a,0} + \bar{\mathrm{A}}) \mathrm{U} \mathbf{s}_{a,i}} = g_1^{\mathrm{B}_a \tilde{\mathrm{A}}_{a,0} \mathrm{U} \mathbf{s}_{a,i} + \pi_3(\mathrm{B}_a \bar{\mathrm{A}})(\mathbf{a} \circ \tilde{\mathbf{s}}_{a,i,3})}$$

for $i \in [w_1]^+$ (corresponding to $b_0 s_i$), which can be computed using $g_1^{\mathbf{a}}$ and $g_1^{\mathbf{a}\circ\mathbf{z}}$. We use $g_1^{\mathrm{B}_a \mathrm{A}_{a,j} \mathrm{U} \mathbf{s}_{a,i}}$ for $i \in [w_1]^+, j \in [n]$ (corresponding to $b_j s_i$), which can be computed using $g_1^{\mathbf{a}}$. Finally, note that we can also construct the value $g_1^{\mathrm{B}_a \mathrm{U} \boldsymbol{\omega}_a}$.

The challenger blinds the value $e(g_1, g_2)^{\delta_a}$ by setting $\mathsf{ct}_{a,0}$ as

$$e(g_1, g_2)^{\delta_a} \cdot e\Big(g_1^{\mathrm{B}_a \mathrm{U} \mathbf{s}_{a,0}}, g_2^{\mathrm{B}_a^* \boldsymbol{\alpha}_a}\Big)$$
$$= e(g_1, g_2)^{\delta_a} \cdot e\Big(g_1^{\mathrm{U} \mathbf{s}_{a,0}}, g_2^{\boldsymbol{\alpha}_a}\Big)$$
$$= e(g_1, g_2)^{\delta_a} \cdot e\Big(g_1^{\rho_1(\pi_1(\mathrm{U}))\mathbf{s}_{a,0,1}}, g_2^{\boldsymbol{\alpha}_{a,1}}\Big) \cdot e\Big(g_1^{\mathbf{s}_{a,0,2}}, g_2^{\boldsymbol{\alpha}_{a,2}}\Big) \cdot e\Big(g_1^{\mathbf{a}\circ\tilde{\mathbf{s}}_{a,0,3}}, g_2^{x\mathbf{z}-\tilde{\boldsymbol{\alpha}}_{a,3}}\Big)$$
$$= e(g_1, g_2)^{\delta_a} \cdot e\Big(g_1^{\rho_1(\pi_1(\mathrm{U}))\mathbf{s}_{a,0,1}}, g_2^{\boldsymbol{\alpha}_{a,1}}\Big) \cdot e\Big(g_1^{\mathbf{s}_{a,0,2}}, g_2^{\boldsymbol{\alpha}_{a,2}}\Big) \cdot e\Big(g_1^{\mathbf{a}\circ\mathbf{z}\circ\tilde{\mathbf{s}}_{a,0,3}}, g_2^{\mathbf{x}}\Big) \cdot e\Big(g_1^{\mathbf{a}\circ\tilde{\mathbf{s}}_{a,0,3}}, g_2^{-\tilde{\boldsymbol{\alpha}}_{a,3}}\Big),$$

which can be computed using $g_1^{\mathbf{a}}$, $g_1^{\mathbf{a}\circ\mathbf{z}}$, and g_2^x.

The complete challenge ciphertext is

$$\mathsf{ct} = \big(\mathsf{ct}_0, \{\mathsf{ct}_{a,0}, \mathsf{ct}_{a,1,0}, \ldots, \mathsf{ct}_{a,1,w_1}, \mathsf{ct}_{a,2,1}, \ldots, \mathsf{ct}_{a,2,w_3}\}_{a \in \mathcal{A}^*}\big).$$

Recall that this challenge ciphertext has the property that

$$\prod_{a \in \mathcal{A}^*} e(g_1, g_2)^{\delta_a} = e(g_1, g_2)^{xy(z_1 + \cdots + z_d)}.$$

So, if $T = e(g_1, g_2)^{xy(z_1+\cdots+z_d)}$, the adversary $\mathcal{A}$ is playing $\mathsf{Game}_{2,q,2}$ and otherwise, if $T \in_R \mathbb{G}_T$, $\mathcal{A}$ is playing Game_3. □

7 Conclusions

In this chapter, we conclude our findings and summarize the main contributions of this dissertation. We also discuss our results and how they improve over the state-of-the-art.

In this dissertation, we have researched the possibility to strengthen the control on data sharing through constructing new cryptographic algorithms. We have established that it is possible to limit the amount of information that is revealed during data sharing by encrypting the data using multi-client functional encryption (MC-FE). Hence, we set ourselves the research goal of developing "fine-grained data protection techniques" in the introduction of this thesis. Central to our research are two research questions: Research Question 1 on how new functionalities can be achieved, Research Question 2 on assessing the efficiency of the proposed schemes.

7.1 Ways of Achieving Special-Purpose MC-FE

Looking at the proposed constructions, we can answer Research Question 1 and also reflect on the necessary cryptographic primitives. We recognize that MC-FE constructions can be based on a broad variety of cryptographic primitives. For operations on two sets, we used a combination of pseudorandom functions (PRFS), secret sharing, and elliptic curve cryptography (ECC). Using these relatively simple primitives, we are already able to build complex functionalities, such as our threshold set intersection scheme. When taking inputs from more than two clients, we require the use of hash functions. The prime use of the hash function is to map a globally unique identifier, $\mathsf{ID} \in \mathcal{ID}$, to a random (group) element so that mix-and-match attacks are prevented. For all our multi-client constructions for set operations, we do not require any other primitives, since building blocks like distributed encryption (DE) can be constructed using the already mentioned primitives [LHK14]. To evaluate predicates using MC-FE, we see that we required pairing-based cryptography (PBC) for both vector equality testing and our compiler for multi-authority predicate encryption (MA-PE).

A natural question is whether we can construct MC-FE by relying on other primitives than we used in our constructions. The most prominent question

we might ask, is whether hash functions, and the use of the random oracle model (ROM) in the corresponding security proofs, are always necessary when we consider a construction for more than two clients. While multi-authority attribute-based encryption (MA-ABE) schemes without relying on the ROM exist [LW11; LCH+11], these schemes suffer from other drawbacks, such as providing no collusion resistance or requiring interaction to obtain a new identifier ID. These limitations make these approaches unsuitable for our requirement of non-interactive MC-FE. Therefore it remains an open question whether MC-FE schemes for more than two clients in the standard model exists.

We note that MC-FE schemes exist that are based on other primitives. For the inner-product functionality, where the function is to compute the value of the inner-products $\sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{y}_i \rangle$ (not to be confused with inner-product predicate encryption (IPPE) where the inner-product serves as a test), there exist constructions that can be based on learning with errors or Paillier's composite residuosity assumption [ABK+19]. Whether these alternative complexity assumptions are also applicable to other MC-FE functionalities is unknown since these functionalities were only recently introduced as an MC-FE scheme.

7.2 Efficiency of MC-FE

To answer Research Question 2 about the efficiency of the constructions, we consider our own implementations of several schemes [CRIPTIM] and evaluations in literature, as far as available. While for a specific use case the efficiency might differ, we give a simplified overview of the (estimated) efficiency of various schemes in Figure 7.1. We cannot report on the efficiency of MC-FE for the inner-product functionality, as no implementations for these schemes are available. Similarly, no general-purpose MC-FE schemes are implemented, but instead we use evaluation results for single client functional encryption (FE).

We see that the (special-purpose) MC-FE constructions for the functionalities of summing and two-client set operations are most efficient. Both constructions are also based on fast and simple cryptographic primitives, such as hash functions and ECC. However, this is not a guarantee that the final construction will be fast as well. For example, the construction for multi-client set operations is also based on the same fast primitives, but is roughly six orders of magnitude slower than their two client counterpart. With an evaluation time within minutes on commodity hardware, these constructions are also efficient enough for use in practice, although their applicability

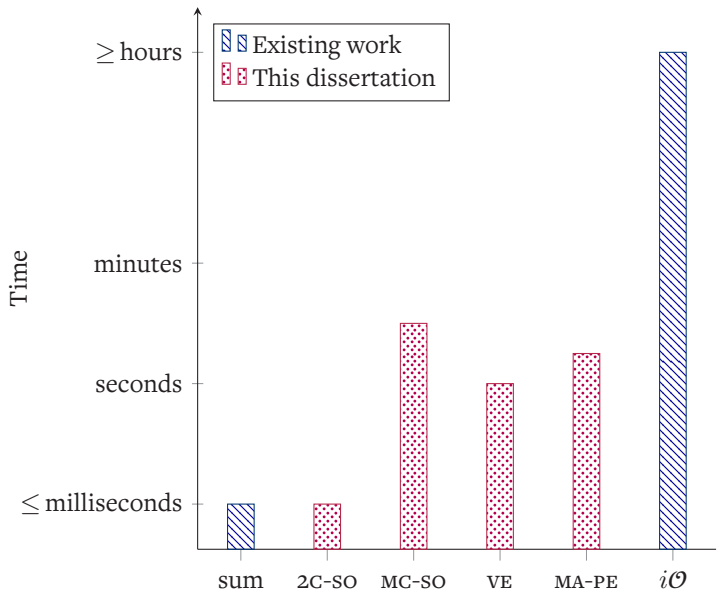


Figure 7.1. Overview of the expected time it takes to evaluate a function using an MC-FE scheme on commodity hardware.
Constructions:
- summing ([SCR+11]);
- SO = Set Operations (Chapter 3);
- VE = Vector Equality (Chapter 4);
- MA-PE = multi-authority predicate encryption (Chapter 5);
- $i\mathcal{O}$ = indistinguishability obfuscation ([BOK+15]).

generally depends on the concrete use case. Indeed, as we explain in the detailed analysis in Section 3.8, the speed of a particular construction greatly depends on the number of clients and the set sizes.

The PBC constructions for predicates, *i.e.*, for vector equality testing or using the MA-PE compiler, should be thought of having a running time in the order of seconds for most use cases. The type of bilinear map that is used has a high influence on the speed of the constructions. Type 3 pairings are typically significantly faster that Type 1 pairings for the same security level: Freeman [Fre10] claims that Type 3 pairings can be about 50 times faster compared to a composite-order Type 1 pairing. The vector equality testing scheme from Chapter 4 uses Type 3 pairings, while the MA-PE schemes from Chapter 5 require a composite-order Type 1 pairing. However, depending on the use case it can be acceptable to wait a few seconds to minutes before obtaining the result of the computation, *e.g.*, when monitoring multiple clients once every hour. In case speed is a considerable bottleneck, Section 6.2 describes a detailed idea on how to use a similar MA-PE compiler on prime-order Type 3 pairings.

From the figure it should be clear that *general-purpose* MC-FE schemes are still far from practical. As explained, no general-purpose MC-FE constructions have been implemented. There have only been some efforts in implementing part of general-purpose FE schemes. Most evaluations [AHK+14; LMA+16] are on program obfuscations of point functions, *i.e.*, a function that outputs TRUE

for one specific, obfuscated (hidden) value and FALSE otherwise. Using a supercomputer with several dozens (32–128) of cores and many gigabytes of RAM (250–2048) these constructions can get down to evaluation times of several minutes for a claimed security level in the range between $\lambda = 52$ and 80 bits. Lewi *et al.* [LMA+16] also implement order-revealing encryption using multilinear maps. While the implemented construction is only multi-input functional encryption (MI-FE), not multi-*client*, and limited to a special-purpose, the evaluation time of a single comparison is still in the order of minutes. For generic functionalities, Halevi *et al.* [HHS+17] implemented small sized branching programs that they could evaluate in the order of minutes, but also concluded that the obfuscation of non-trivial functions "is still extremely limited." Banescu *et al.* [BOK+15] even estimated that it would take 1.3×10^8 *years* for their unoptimized implementation of $i\mathcal{O}$ to evaluate the relatively simple, single client, FE functionality of a 2-bit multiplication circuit on a 2.6 GHz CPU. Realizing that these results are for single-client FE and that MC-FE implementations are presumably even complexer and also slower, we conclude that general-purpose MC-FE is still far from practical.

Compared to general-purpose solutions, our special-purpose MC-FE are a significant step forward in achieving controlled data sharing and effective data sharing in a fine-grained and secure way. Depending on the use case, our constructions are suitable to be applied in a practical scenario as they are efficient and require only conventional and explicitly stated security assumptions.

Bibliography

Author References

[CRIPTIM] T. R. van de Kamp and M. H. Everts. *Implementations of Controlled Data Sharing Schemes*. CRIPTIM consortium. URL: https://github.com/CRIPTIM/.

[KPE+16] T. R. van de Kamp, A. Peter, M. H. Everts, and W. Jonker. "Private Sharing of IOCs and Sightings." In: *Workshop on Information Sharing and Collaborative Security* (WISCS). Ed. by E.-O. Blass and F. Kerschbaum. ACM, Oct. 2016, pp. 35–38. DOI: 10.1145/2994539.2994544.

[KPE+17] T. R. van de Kamp, A. Peter, M. H. Everts, and W. Jonker. "Multi-client Predicate-Only Encryption for Conjunctive Equality Tests." In: *Cryptology And Network Security* (CANS). Ed. by S. Capkun and S. S. M. Chow. Vol. 11261. LNCS. Springer, 2017, pp. 135–157. DOI: 10.1007/978-3-030-02641-7_7.

[KPJ20] T. R. van de Kamp, A. Peter, and W. Jonker. "A Multi-authority Approach to Various Predicate Encryption Types." In: *Designs, Codes and Cryptography* (DESI) 88.2 (Feb. 2020), pp. 363–390. DOI: 10.1007/s10623-019-00686-x.

[KSJ+19] T. R. van de Kamp, D. Stritzl, W. Jonker, and A. Peter. "Two-Client and Multi-client Functional Encryption for Set Intersection." In: *Australasian Conference on Information Security and Privacy* (ACISP). Ed. by J. Jang-Jaccard and F. Guo. Vol. 11547. LNCS. Springer, 2019, pp. 97–115. DOI: 10.1007/978-3-030-21548-4_6.

Other References

[ABK+19] M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. "Decentralizing Inner-Product Functional Encryption." In: *Public-Key Cryptography* (PKC). Ed. by D. Lin and K. Sako. Vol. 11443.II. LNCS. Springer, 2019, pp. 128–157. DOI: 10.1007/978-3-030-17259-6_5.

[ABS17] M. Ambrona, G. Barthe, and B. Schmidt. "Generic Transformations of Predicate Encodings: Constructions and Applications." In: CRYPTO. Ed. by J. Katz and H. Shacham. Vol. 10401.I. LNCS. Springer, 2017, pp. 36–66. DOI: 10.1007/978-3-319-63688-7_2.

[AC15] S. Agrawal and M. Chase. *A Study of Pair Encodings: Predicate Encryption in Prime Order Groups*. Tech. rep. Mar. 2017 (May 1, 2015). IACR: 2015/413.

[AC16] S. Agrawal and M. Chase. "A Study of Pair Encodings: Predicate Encryption in Prime Order Groups." In: *Theory of Cryptography* (TCC). Ed. by E. Kushilevitz and T. Malkin. Vol. 9563.II. LNCS. Springer, 2016, pp. 259–288. DOI: 10.1007/978-3-662-49099-0_10.

[AC17] S. Agrawal and M. Chase. "Simplifying Design and Analysis of Complex Predicate Encryption Schemes." In: EUROCRYPT. Ed. by J.-S. Coron and J. B. Nielsen. Vol. 10210.I. LNCS. Springer, 2017, pp. 627–656. DOI: 10.1007/978-3-319-56620-7_22.

[ACD+06] M. Abdalla, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven, and N. P. Smart. "Identity-Based Encryption Gone Wild." In: *International Colloquium on Automata, Languages, and Programming* (ICALP). Ed. by M. Bugliesi *et al.* Vol. 4052.II. LNCS. Springer, 2006, pp. 300–311. DOI: 10.1007/11787006_26.

[AGM+13] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin. "Charm: a framework for rapidly prototyping cryptosystems." In: *Journal of Cryptographic Engineering* (JCEN) 3.2 (June 2013). Ed. by Ç. K. Koç, pp. 111–128. DOI: 10.1007/s13389-013-0057-3. URL: https://github.com/JHUISI/charm.

[AGR+17] M. Abdalla, R. Gay, M. Raykova, and H. Wee. "Multi-Input Inner-Product Functional Encryption from Pairings." In: EUROCRYPT. Ed. by J.-S. Coron and J. B. Nielsen. Vol. 10210.I. LNCS. Springer, 2017, pp. 601–626. DOI: 10.1007/978-3-319-56620-7_21.

[AHK+14] D. Apon, Y. Huang, J. Katz, and A. J. Malozemoff. *Implementing Cryptographic Program Obfuscation*. Tech. rep. Feb. 2015 (Oct. 1, 2014). IACR: 2014/779.

[AI09] N. Attrapadung and H. Imai. "Dual-Policy Attribute Based Encryption." In: *Applied Cryptography and Network Security* (ACNS). Ed. by M. Abdalla *et al.* Vol. 5536. LNCS. Springer, 2009, pp. 168–185. DOI: 10.1007/978-3-642-01957-9_11.

[ATD15] A. Abadi, S. Terzis, and C. Dong. "O-PSI: Delegated Private Set Intersection on Outsourced Datasets." In: *ICT Systems Security and Privacy Protection* (SEC). Ed. by H. Federrath and D. Gollmann. Vol. 455. IFIP AICT. Springer, 2015, pp. 3–17. DOI: 10.1007/978-3-319-18467-8_1.

[ATD16] A. Abadi, S. Terzis, and C. Dong. "VD-PSI: Verifiable Delegated Private Set Intersection on Outsourced Private Datasets." In: *Financial Cryptography and Data Security* (FC). Ed. by J. Grossklags and B. Preneel. Vol. 9603. LNCS. Springer, 2016, pp. 149–168. DOI: 10.1007/978-3-662-54970-4_9.

[Att14] N. Attrapadung. "Dual System Encryption via Doubly Selective Security: Framework, Fully Secure Functional Encryption for Regular Languages, and More." In: EUROCRYPT. Ed. by P. Q. Nguyen and E. Oswald. Vol. 8441. LNCS. Springer, 2014, pp. 557–577. DOI: 10.1007/978-3-642-55220-5_31.

[AY15] N. Attrapadung and S. Yamada. "Duality in ABE: Converting Attribute Based Encryption for Dual Predicate and Dual Policy via Computational Encodings." In: *Cryptographers' Track at the RSA Conference* (CT-RSA). Ed. by K. Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 87–105. DOI: 10.1007/978-3-319-16715-2_5.

[BB04a] D. Boneh and X. Boyen. *Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles*. Tech. rep. Dec. 2004 (July 20, 2004). IACR: 2004/172.

[BB04b] D. Boneh and X. Boyen. "Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles." In: EUROCRYPT. Ed. by C. Cachin and J. L. Camenisch. Vol. 3027. LNCS. Springer, 2004, pp. 223–238. DOI: 10.1007/978-3-540-24676-3_14.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. "Short Group Signatures." In: CRYPTO. Ed. by M. Franklin. Vol. 3152. LNCS. Springer, 2004, pp. 41–55. DOI: 10.1007/978-3-540-28628-8_3.

[BD18] R. Barbulescu and S. Duquesne. "Updating Key Size Estimations for Pairings." In: *Journal of Cryptology* (J CRYPTOL) (Jan. 2018). Ed. by K. G. Paterson. DOI: 10.1007/s00145-018-9280-5.

[BF01] D. Boneh and M. Franklin. "Identity-Based Encryption from the Weil Pairing." In: CRYPTO. Ed. by J. Kilian. Vol. 2139. LNCS. Springer, 2001, pp. 213–229. DOI: 10.1007/3-540-44647-8_13.

[BGN05] D. Boneh, E.-J. Goh, and K. Nissim. "Evaluating 2-DNF Formulas on Ciphertexts." In: *Theory of Cryptography* (TCC). Ed. by J. Kilian. Vol. 3378. LNCS. Springer, 2005, pp. 325–341. DOI: 10.1007/978-3-540-30576-7_18.

[BHJ+14] C. Bösch, P. Hartel, W. Jonker, and A. Peter. "A Survey of Provably Secure Searchable Encryption." In: *ACM Computing Surveys* (CSUR) 47.2 (Aug. 2014). Ed. by S. Sahni, 18:1–18:51. DOI: 10.1145/2636328.

[BHJ+19] B. Barak, S. B. Hopkins, A. Jain, P. Kothari, and A. Sahai. "Sum-of-Squares Meets Program Obfuscation, Revisited." In: EUROCRYPT. Ed. by Y. Ishai and V. Rijmen. Vol. 11476.I. LNCS. Springer, 2019, pp. 226–250. DOI: 10.1007/978-3-030-17653-2_8.

[BKR13] M. Bellare, S. Keelveedhi, and T. Ristenpart. "Message-Locked Encryption and Secure Deduplication." In: EUROCRYPT. Ed. by T. Johansson and P. Q. Nguyen. Vol. 7881. LNCS. Springer, 2013, pp. 296–312. DOI: 10.1007/978-3-642-38348-9_18.

[Blo70] B. H. Bloom. "Space/Time Trade-offs in Hash Coding with Allowable Errors." In: *Communications of the ACM* (COMMUN. ACM) 13.7 (July 1970). Ed. by M. Stuart Lynn, pp. 422–426. DOI: 10.1145/362686.362692.

[BLR+15] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. "Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation." In: EUROCRYPT. Ed. by E. Oswald and M. Fischlin. Vol. 9057.II. LNCS. Springer, 2015, pp. 563–594. DOI: 10.1007/978-3-662-46803-6_19.

[BN00] M. Bellare and C. Namprempre. "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm." In: ASIACRYPT. Ed. by T. Okamoto. Vol. 1976. LNCS. Springer, 2000, pp. 531–545. DOI: 10.1007/3-540-44448-3_41.

[BN06] P. S. L. M. Barreto and M. Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order." In: *Selected Areas in Cryptography* (SAC). Ed. by B. Preneel and S. Tavares. Vol. 3897. LNCS. Springer, 2006, pp. 319–331. DOI: 10.1007/11693383_22.

[BOK+15] S. Banescu, M. Ochoa, N. Kunze, and A. Pretschner. "Idea: Benchmarking Indistinguishability Obfuscation – A Candidate Implementation." In: *Engineering Secure Software and Systems* (ESSOS). Ed. by F. Piessens *et al.* Vol. 8978. LNCS. Springer, 2015, pp. 149–156. DOI: 10.1007/978-3-319-15618-7_12.

[BS15] Z. Brakerski and G. Segev. "Function-Private Functional Encryption in the Private-Key Setting." In: *Theory of Cryptography* (TCC). Ed. by Y. Dodis and J. B. Nielsen. Vol. 9015. LNCS. Springer, 2015, pp. 306–324. DOI: 10.1007/978-3-662-46497-7_12.

[BSW11] D. Boneh, A. Sahai, and B. Waters. "Functional Encryption: Definitions and Challenges." In: *Theory of Cryptography* (TCC). Ed. by Y. Ishai. Vol. 6597. LNCS. Springer, 2011, pp. 253–273. DOI: 10.1007/978-3-642-19571-6_16.

[BSW13] K. Benson, H. Shacham, and B. Waters. "The k-BDH Assumption Family: Bilinear Map Cryptography from Progressively Weaker Assumptions." In: *Cryptographers' Track at the RSA Conference* (CT-RSA). Ed. by E. Dawson. Vol. 7779. LNCS. Springer, 2013, pp. 310–325. DOI: 10.1007/978-3-642-36095-4_20.

[BW06] X. Boyen and B. Waters. "Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles)." In: CRYPTO. Ed. by C. Dwork. Springer, 2006, pp. 290–307. DOI: 10.1007/11818175_17.

[BW07] D. Boneh and B. Waters. "Conjunctive, Subset, and Range Queries on Encrypted Data." In: *Theory of Cryptography* (TCC). Ed. by S. P. Vadhan. Vol. 4392. LNCS. Springer, 2007, pp. 535–554. DOI: 10.1007/978-3-540-70936-7_29.

[BWY11] M. Bellare, B. Waters, and S. Yilek. "Identity-Based Encryption Secure against Selective Opening Attack." In: *Theory of Cryptography* (TCC). Ed. by Y. Ishai. Vol. 6597. LNCS. Springer, 2011, pp. 235–252. DOI: 10.1007/978-3-642-19571-6_15.

[BZ06] M. Barbaro and T. Zeller Jr. *A Face Is Exposed for AOL Searcher No. 4417749*. The New York Times Company. Aug. 9, 2006. URL: https://nyti.ms/2k4BTyC (visited on 2019-05-01).

[CC09] M. Chase and S. S. M. Chow. "Improving Privacy and Security in Multi-authority Attribute-based Encryption." In: *Conference on Computer and Communications Security* (CCS). Ed. by E. Al-Shaer *et al.* ACM, 2009, pp. 121–130. DOI: 10.1145/1653662.1653678.

[CDG+18] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. "Decentralized Multi-Client Functional Encryption for Inner Product." In: ASIACRYPT. Ed. by T. Peyrin and S. Galbraith. Vol. 11273. LNCS. Springer, 2018, pp. 703–732. DOI: 10.1007/978-3-030-03329-3_24.

[CFS+17] X. Carpent, S. Faber, T. Sander, and G. Tsudik. "Private Set Projections & Variants." In: *Workshop on Privacy in the Electronic Society* (WPES). Ed. by A. J. Lee. ACM, 2017, pp. 87–98. DOI: 10.1145/3139550.3139554.

[CGH17] Y. Chen, C. Gentry, and S. Halevi. "Cryptanalyses of Candidate Branching Program Obfuscators." In: EUROCRYPT. Ed. by J.-S. Coron and J. B. Nielsen. Vol. 10212.III. LNCS. Springer, 2017, pp. 278–307. DOI: 10.1007/978-3-319-56617-7_10.

[CGW15] J. Chen, R. Gay, and H. Wee. "Improved Dual System ABE in Prime-Order Groups via Predicate Encodings." In: EUROCRYPT. Ed. by E. Oswald and M. Fischlin. Vol. 9057.II. LNCS. Springer, 2015, pp. 595–624. DOI: 10.1007/978-3-662-46803-6_20.

[Cha07] M. Chase. "Multi-authority Attribute Based Encryption." In: *Theory of Cryptography* (TCC). Ed. by S. P. Vadhan. Vol. 4392. LNCS. Springer, 2007, pp. 515–534. DOI: 10.1007/978-3-540-70936-7_28.

[CHL+15] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. "Cryptanalysis of the Multilinear Map over the Integers." In: EUROCRYPT. Ed. by E. Oswald and M. Fischlin. Vol. 9056.I. LNCS. Springer, 2015, pp. 3–12. DOI: 10.1007/978-3-662-46800-5_1.

[CLL+16] J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. "Cryptanalysis of GGH15 Multilinear Maps." In: CRYPTO. Ed. by M. Robshaw and J. Katz. Vol. 9815.II. LNCS. Springer, 2016, pp. 607–628. DOI: 10.1007/978-3-662-53008-5_21.

[CLO+06] S. H. Conrad, R. J. LeClaire, G. P. O'Reilly, and H. Uzunalioglu. "Critical national infrastructure reliability modeling and analysis." In: *Bell Labs Technical Journal* 11.3 (2006). Ed. by C. Bahr, pp. 57–71. DOI: 10.1002/bltj.20178.

[CLW+16] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. "Practical Order-Revealing Encryption with Limited Leakage." In: *Fast Software Encryption* (FSE). Ed. by T. Peyrin. Vol. 9783. LNCS. Springer, 2016, pp. 474–493. DOI: 10.1007/978-3-662-52993-5_24.

[CMZ14] M. Chase, S. Meiklejohn, and G. Zaverucha. “Algebraic MACs and Keyed-Verification Anonymous Credentials.” In: *Computer and Communications Security* (CCS). Ed. by P. Ning *et al.* ACM, 2014, pp. 1205–1216. DOI: 10.1145/2660267.2660328.

[CW13] J. Chen and H. Wee. “Fully, (Almost) Tightly Secure IBE and Dual System Groups.” In: CRYPTO. Ed. by R. Canetti and J. A. Garay. Vol. 8043.II. LNCS. Springer, 2013, pp. 435–460. DOI: 10.1007/978-3-642-40084-1_25.

[CW14] J. Chen and H. Wee. *Dual System Groups and its Applications – Compact HIBE and More*. Tech. rep. Apr. 2014. IACR: 2014/265.

[DAB+02] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. “Reclaiming space from duplicate files in a serverless distributed file system.” In: *International Conference on Distributed Computing Systems* (ICDCS). Ed. by L. E. T. Rodrigues *et al.* IEEE, 2002, pp. 617–624. DOI: 10.1109/ICDCS.2002.1022312.

[DCW13] C. Dong, L. Chen, and Z. Wen. “When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol.” In: *Computer & Communications Security* (CCS). Ed. by A.-R. Sadeghi *et al.* ACM, 2013, pp. 789–800. DOI: 10.1145/2508859.2516701.

[DK11] B. Ding and A. C. König. “Fast Set Intersection in Memory.” In: *Proceedings of the VLDB Endowment* (PVLDB) 4.4 (Jan. 2011). Ed. by H. V. Jagadish and N. Koudas, pp. 255–266. DOI: 10.14778/1938545.1938550.

[DOT18] P. Datta, T. Okamoto, and J. Tomida. “Full-Hiding (Unbounded) Multi-input Inner Product Functional Encryption from the k-Linear Assumption.” In: *Public Key Cryptography* (PKC). Ed. by M. Abdalla and R. Dahab. Vol. 10770.II. LNCS. Springer, 2018, pp. 245–277. DOI: 10.1007/978-3-319-76581-5_9.

[DS09] M. Dunn-Cavelty and M. Suter. “Public–Private Partnerships are no silver bullet: An expanded governance model for Critical Infrastructure Protection.” In: *International Journal of Critical Infrastructure Protection* 2.4 (2009). Ed. by S. Shenoi, pp. 179–187. DOI: 10.1016/j.ijcip.2009.08.006.

[DT10] E. De Cristofaro and G. Tsudik. “Practical Private Set Intersection Protocols with Linear Complexity.” In: *Financial Cryptography and Data Security* (FC). Ed. by R. Sion. Vol. 6052. LNCS. Springer, 2010, pp. 143–159. DOI: 10.1007/978-3-642-14577-3_13.

[EULex] “Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union.” *Official Journal of the European Union* (OJ) L194, July 2016, pp. 1–30. ELI: 2016/1148.

[FM19] G. Fotiadis and C. Martindale. *Optimal TNFS-secure pairings on elliptic curves with composite embedding degree*. Tech. rep. May 2019. IACR: 2019/555.

[FNP04] M. J. Freedman, K. Nissim, and B. Pinkas. "Efficient Private Matching and Set Intersection." In: EUROCRYPT. Ed. by C. Cachin and J. L. Camenisch. Vol. 3027. LNCS. Springer, 2004, pp. 1–19. DOI: 10.1007/978-3-540-24676-3_1.

[Fre10] D. M. Freeman. "Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups." In: EUROCRYPT. Ed. by H. Gilbert. Vol. 6110. LNCS. Springer, 2010, pp. 44–61. DOI: 10.1007/978-3-642-13190-5_3.

[GGG+14] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. "Multi-input Functional Encryption." In: EUROCRYPT. Ed. by P. Q. Nguyen and E. Oswald. Vol. 8441. LNCS. Springer, 2014, pp. 578–602. DOI: 10.1007/978-3-642-55220-5_32.

[GGH+13a] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits." In: *Foundations of Computer Science* (FOCS). Ed. by O. Reingold. 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13.

[GGH+13b] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. "Attribute-Based Encryption for Circuits from Multilinear Maps." In: CRYPTO. Ed. by R. Canetti and J. A. Garay. Vol. 8043.II. LNCS. Springer, 2013, pp. 479–499. DOI: 10.1007/978-3-642-40084-1_27.

[GH11] D. Galindo and J.-H. Hoepman. "Non-interactive Distributed Encryption: A New Primitive for Revocable Privacy." In: *Workshop on Privacy in the Electronic Society* (WPES). Ed. by Y. Chen and J. Vaidya. ACM, 2011, pp. 81–92. DOI: 10.1145/2046556.2046567.

[GKL+13] S. D. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. *Multi-Input Functional Encryption*. Tech. rep. Nov. 2013. IACR: 2013/774.

[GKP+13] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. "How to Run Turing Machines on Encrypted Data." In: CRYPTO. Ed. by R. Canetti and J. A. Garay. Vol. 8043.II. LNCS. Springer, 2013, pp. 536–553. DOI: 10.1007/978-3-642-40084-1_30.

[GM84] S. Goldwasser and S. Micali. "Probabilistic Encryption." In: *Journal of Computer and System Sciences* 28.2 (1984). Ed. by H. R. A. Lewis, pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9.

[Goo13] D. Goodin. *Crypto weakness in Web comment system exposes hate-mongering politicians. Journalists exploit weakness in Gravatar to identify extremist forum members*. Condé Nast. Dec. 11, 2013. URL: https://arstechnica.com/?post_type=post&p=383933 (visited on 2019-05-01).

[GP09] P. Golle and K. Partridge. “On the Anonymity of Home/Work Location Pairs.” In: PERVASIVE. Ed. by H. Tokuda *et al.* Vol. 5538. LNCS. Springer, 2009, pp. 390–397. DOI: 10.1007/978-3-642-01516-8_26.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. “Pairings for cryptographers.” In: *Discrete Applied Mathematics* 156.16 (2008). Ed. by M. I. González Vasco and R. Steinwandt. Applications of Algebra to Cryptography, pp. 3113–3121. DOI: 10.1016/j.dam.2007.12.010.

[HHS+17] S. Halevi, T. Halevi, V. Shoup, and N. Stephens-Davidowitz. “Implementing BP-Obfuscation Using Graph-Induced Encoding.” In: *Computer and Communications Security* (CCS). Ed. by B. Thuraisingham *et al.* ACM, 2017, pp. 783–798. DOI: 10.1145/3133956.3133976.

[IKN+19] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, M. Raykova, S. Saxena, K. Seth, D. Shanahan, and M. Yung. *On Deploying Secure Computing Commercially: Private Intersection-Sum Protocols and their Business Applications*. Tech. rep. June 2019. IACR: 2019/723.

[IRTPA] “Intelligence Reform and Terrorism Prevention Act of 2004,” Dec. 2004. URL: https://www.gpo.gov/fdsys/pkg/STATUTE-118/pdf/STATUTE-118-Pg3638.pdf (visited on 2019-03-26).

[JL10] S. Jarecki and X. Liu. “Fast Secure Computation of Set Intersection.” In: *Security and Cryptography for Networks* (SCN). Ed. by J. A. Garay and R. De Prisco. Vol. 6280. LNCS. Springer, 2010, pp. 418–435. DOI: 10.1007/978-3-642-15317-4_26.

[Ker11] F. Kerschbaum. “Public-Key Encrypted Bloom Filters with Applications to Supply Chain Integrity.” In: *Data and Applications Security and Privacy* (DBSEC). Ed. by Y. Li. Vol. 6818. LNCS. Springer, 2011, pp. 60–75. DOI: 10.1007/978-3-642-22348-8_7.

[Ker12a] F. Kerschbaum. “Collusion-resistant Outsourcing of Private Set Intersection.” In: *Symposium on Applied Computing* (SAC). Ed. by S. Ossowski and P. Lecca. ACM, 2012, pp. 1451–1456. DOI: 10.1145/2245276.2232008.

[Ker12b] F. Kerschbaum. “Outsourced Private Set Intersection Using Homomorphic Encryption.” In: *Information, Computer and Communications Security* (ASIACCS). Ed. by H. Y. Youm and Y. Won. ACM, 2012. DOI: 10.1145/2414456.2414506.

[KLM+18] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu. “Function-Hiding Inner Product Encryption Is Practical.” In: *Security and Cryptography for Networks* (SCN). Ed. by D. Catalano and R. De Prisco. Vol. 11035. LNCS. Springer, 2018, pp. 544–562. DOI: 10.1007/978-3-319-98113-0_29.

[KMR+14] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. “Scaling Private Set Intersection to Billion-Element Sets.” In: *Financial Cryptography and Data Security* (FC). Ed. by N. Christin and R. Safavi-Naini. Vol. 8437. LNCS. Springer, 2014, pp. 195–215. DOI: 10.1007/978-3-662-45472-5_13.

[KS05] L. Kissner and D. Song. "Privacy-Preserving Set Operations." In: CRYPTO. Ed. by V. Shoup. Vol. 3621. LNCS. Springer, 2005, pp. 241–257. DOI: 10.1007/11535218_15.

[KSG+16] J. Kim, W. Susilo, F. Guo, and M. H. Au. "A Tag Based Encoding: An Efficient Encoding for Predicate Encryption in Prime Order Groups." In: *Security and Cryptography for Networks* (SCN). Ed. by V. Zikas and R. De Prisco. Vol. 9841. LNCS. Springer, 2016, pp. 3–22. DOI: 10.1007/978-3-319-44618-9_1.

[KSW08] J. Katz, A. Sahai, and B. Waters. "Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products." In: EUROCRYPT. Ed. by N. Smart. Vol. 4965. LNCS. Springer, 2008, pp. 146–162. DOI: 10.1007/978-3-540-78967-3_9.

[LCH+11] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen. "Fully Secure Multi-authority Ciphertext-Policy Attribute-Based Encryption without Random Oracles." In: *European Symposium on Research in Computer Security* (ESORICS). Ed. by V. Atluri and C. Diaz. Vol. 6879. LNCS. Springer, 2011, pp. 278–297. DOI: 10.1007/978-3-642-23822-2_16.

[Lew12] A. Lewko. "Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting." In: EUROCRYPT. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. LNCS. Springer, 2012, pp. 318–335. DOI: 10.1007/978-3-642-29011-4_20.

[LHK14] W. Lueks, J.-H. Hoepman, and K. Kursawe. "Forward-Secure Distributed Encryption." In: *Privacy Enhancing Technologies Symposium* (PETS). Ed. by E. De Cristofaro and S. J. Murdoch. Vol. 8555. LNCS. Springer, 2014, pp. 123–142. DOI: 10.1007/978-3-319-08506-7_7.

[LK15a] E. Luiijf and A. Kernkamp. *Sharing Cyber Security Information. Good Practice Stemming from the Dutch Public-Private-Participation Approach*. Tech. rep. Mar. 2015. URL: https://publications.tno.nl/publication/34616508/oLyfG9/luiijf-2015-sharing.pdf (visited on 2019-04-02).

[LK15b] E. Luiijf and M. Klaver. "On the Sharing of Cyber Security Information." In: *International Conference on Critical Infrastructure Protection* (ICCIP). Ed. by M. Rice and S. Shenoi. Vol. 466. IFIPAICT. Springer, 2015, pp. 29–46. DOI: 10.1007/978-3-319-26567-4_3.

[LMA+16] K. Lewi, A. J. Malozemoff, D. Apon, B. Carmer, A. Foltzer, D. Wagner, D. W. Archer, D. Boneh, J. Katz, and M. Raykova. "5Gen: A Framework for Prototyping Applications Using Multilinear Maps and Matrix Branching Programs." In: *Conference on Computer and Communications Security* (CCS). ACM, 2016, pp. 981–992. DOI: 10.1145/2976749.2978376.

[LNK+09] E. Luiijf, A. Nieuwenhuijs, M. Klaver, M. van Eeten, and E. Cruz. "Empirical Findings on Critical Infrastructure Dependencies in Europe." In: *Critical Information Infrastructure Security* (CRITIS). Ed. by R. Setola and S. Geretshuber. Vol. 5508. LNCS. Springer, 2009, pp. 302–310. DOI: 10.1007/978-3-642-03552-4_28.

[LNZ+14] F. Liu, W. K. Ng, W. Zhang, D. H. Giang, and S. Han. "Encrypted Set Intersection Protocol for Outsourced Datasets." In: *International Conference on Cloud Engineering* (IC2E). Ed. by A. Bestavros *et al.* IEEE, 2014, pp. 135–140. DOI: 10.1109/IC2E.2014.18.

[LOS+10] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption." In: EUROCRYPT. Ed. by H. Gilbert. Vol. 6110. LNCS. Springer, 2010, pp. 62–91. DOI: 10.1007/978-3-642-13190-5_4.

[LW10] A. Lewko and B. Waters. "New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts." In: *Theory of Cryptography* (TCC). Ed. by D. Micciancio. Vol. 5978. LNCS. Springer, 2010, pp. 455–479. DOI: 10.1007/978-3-642-11799-2_27.

[LW11] A. Lewko and B. Waters. "Decentralizing Attribute-Based Encryption." In: EUROCRYPT. Ed. by K. G. Paterson. Vol. 6632. LNCS. Springer, 2011, pp. 568–588. DOI: 10.1007/978-3-642-20465-4_31.

[LW12] A. Lewko and B. Waters. "New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques." In: CRYPTO. Ed. by R. Safavi-Naini and R. Canetti. Vol. 7417. LNCS. Springer, 2012, pp. 180–198. DOI: 10.1007/978-3-642-32009-5_12.

[LW16] K. Lewi and D. J. Wu. "Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds." In: *Conference on Computer and Communications Security* (CCS). Ed. by E. R. Weippl *et al.* ACM, 2016. DOI: 10.1145/2976749.2978376.

[Mea86] C. Meadows. "A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party." In: *Security and Privacy* (S&P). Ed. by C. Weissman *et al.* IEEE, 1986, pp. 134–134. DOI: 10.1109/SP.1986.10022.

[MJ18] Y. Michalevsky and M. Joye. "Decentralized Policy-Hiding ABE with Receiver Privacy." In: *European Symposium on Research in Computer Security* (ESORICS). Ed. by J. Lopez *et al.* Vol. 11099.II. LNCS. Springer, 2018, pp. 548–567. DOI: 10.1007/978-3-319-98989-1_27.

[MKE09] S. Müller, S. Katzenbeisser, and C. Eckert. "Distributed Attribute-Based Encryption." In: *Information Security and Cryptology* (ICISC). Ed. by P. J. Lee and J. H. Cheon. Vol. 5461. LNCS. Springer, 2009, pp. 20–36. DOI: 10.1007/978-3-642-00730-9_2.

[MNT01] A. Miyaji, M. Nakabayashi, and S. Takano. "Characterization of Elliptic Curve Traces Under FR-Reduction." In: *International Conference on Information Security and Cryptology* (ICISC). Ed. by D. Won. Vol. 2015. LNCS. Springer, 2001, pp. 90–108. DOI: 10.1007/3-540-45247-8_8.

[MS02] J. D. Moteff and G. M. Stevens. *Critical Infrastructure Information Disclosure and Homeland Security*. Report. Library of Congress Washington DC Congressional Research Service, Aug. 31, 2002. 23 pp. URL: http://www.dtic.mil/docs/citations/ADA467310 (visited on 2017-11-01).

[NR04] M. Naor and O. Reingold. "Number-theoretic Constructions of Efficient Pseudo-random Functions." In: *Journal of the ACM* (JACM) 51.2 (Mar. 2004). Ed. by V. Vianu, pp. 231–262. DOI: 10.1145/972639.972643.

[ONe10] A. O'Neill. *Definitional Issues in Functional Encryption*. Tech. rep. Mar. 2011 (Oct. 30, 2010). IACR: 2010/556.

[OT13] T. Okamoto and K. Takashima. "Decentralized Attribute-Based Signatures." In: *Public-Key Cryptography* (PKC). Ed. by K. Kurosawa and G. Hanaoka. Vol. 7778. LNCS. Springer, 2013, pp. 125–142. DOI: 10.1007/978-3-642-36362-7_9.

[PCC97] President's Commission on Critical Infrastructure Protection. *Critical Foundations: Protecting America's Infrastructures*. Report. Oct. 1997. URL: https://www.fas.org/sgp/library/pccip.pdf (visited on 2016-05-01).

[PR12] O. Pandey and Y. Rouselakis. "Property Preserving Symmetric Encryption." In: EUROCRYPT. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. LNCS. Springer, 2012, pp. 375–391. DOI: 10.1007/978-3-642-29011-4_23.

[PSZ14] B. Pinkas, T. Schneider, and M. Zohner. "Faster Private Set Intersection Based on OT Extension." In: USENIX SECURITY. Ed. by K. Fu and J. Jung. USENIX Association, 2014, pp. 797–812. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas (visited on 2019-01-30).

[RW15] Y. Rouselakis and B. Waters. "Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption." In: *Financial Cryptography and Data Security* (FC). Ed. by R. Böhme and T. Okamoto. Springer, 2015, pp. 315–332. DOI: 10.1007/978-3-662-47854-7_19.

[Sch80] J. T. Schwartz. "Fast Probabilistic Algorithms for Verification of Polynomial Identities." In: *Journal of the ACM* (JACM) 27.4 (Oct. 1980). Ed. by M. R. Garey, pp. 701–717. DOI: 10.1145/322217.322225.

[SCR+11] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song. "Privacy-Preserving Aggregation of Time-Series Data." In: *Network and Distributed System Security Symposium* (NDSS). The Internet Society, 2011. URL: https://www.ndss-symposium.org/ndss2011/privacy-preserving-aggregation-of-time-series-data/ (visited on 2019-01-30).

[Sha07] H. Shacham. *A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants*. Tech. rep. Apr. 2009 (Feb. 25, 2007). IACR: 2007/074.

[Sha79] A. Shamir. “How to Share a Secret.” In: *Communications of the ACM* (COMMUN. ACM) 22.11 (Nov. 1979). Ed. by R. L. Rivest, pp. 612–613. DOI: 10.1145/359168.359176.

[Sho97] V. Shoup. “Lower Bounds for Discrete Logarithms and Related Problems.” In: EUROCRYPT. Ed. by W. Fumy. Vol. 1233. LNCS. Springer, 1997, pp. 256–266. DOI: 10.1007/3-540-69053-0_18.

[Sma01] N. P. Smart. “The Exact Security of ECIES in the Generic Group Model.” In: *IMA International Conference on Cryptography and Coding* (IMACC). Ed. by B. Honary. Vol. 2260. LNCS. Springer, 2001, pp. 73–84. DOI: 10.1007/3-540-45325-3_8.

[SSF16] F. Skopik, G. Settanni, and R. Fiedler. “A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing.” In: *Computers & Security* 60 (July 2016). Ed. by E. H. Spafford, pp. 154–176. DOI: 10.1016/j.cose.2016.04.003.

[SSW09] E. Shen, E. Shi, and B. Waters. “Predicate Privacy in Encryption Systems.” In: *Theory of Cryptography* (TCC). Ed. by O. Reingold. Vol. 5444. LNCS. Springer, 2009, pp. 457–473. DOI: 10.1007/978-3-642-00457-5_27.

[SW05] A. Sahai and B. Waters. “Fuzzy Identity-Based Encryption.” In: EUROCRYPT. Ed. by R. Cramer. Vol. 3494. LNCS. Springer, 2005, pp. 457–473. DOI: 10.1007/11426639_27.

[Tro14] J. K. Trotter. *Public NYC Taxicab Database Lets You See How Celebrities Tip*. Gawker Media Group. Oct. 23, 2014. URL: https://gawker.com/the-public-nyc-taxicab-database-that-accidentally-track-1646724546 (visited on 2019-05-01).

[Wat09] B. Waters. “Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions.” In: CRYPTO. Ed. by S. Halevi. Vol. 5677. LNCS. Springer, 2009, pp. 619–636. DOI: 10.1007/978-3-642-03356-8_36.

[Wat12] B. Waters. “Functional Encryption for Regular Languages.” In: CRYPTO. Ed. by R. Safavi-Naini and R. Canetti. Vol. 7417. LNCS. Springer, 2012, pp. 218–235. DOI: 10.1007/978-3-642-32009-5_14.

[Wee14] H. Wee. “Dual System Encryption via Predicate Encodings.” In: *Theory of Cryptography* (TCC). Ed. by Y. Lindell. Vol. 8349. LNCS. Springer, 2014, pp. 616–637. DOI: 10.1007/978-3-642-54242-8_26.

[YTH+10] G. Yang, C. H. Tan, Q. Huang, and D. S. Wong. “Probabilistic Public Key Encryption with Equality Test.” In: *Cryptographers’ Track at the RSA Conference* (CT-RSA). Ed. by J. Pieprzyk. Vol. 5985. LNCS. Springer, 2010, pp. 119–131. DOI: 10.1007/978-3-642-11925-5_9.

[ZX15] Q. Zheng and S. Xu. "Verifiable Delegated Set Intersection Operations on Outsourced Encrypted Data." In: *International Conference on Cloud Engineering* (IC2E). Ed. by K. S. Candan and K. D. Ryu. IEEE, 2015, pp. 175–184. DOI: 10.1109/IC2E.2015.38.

Colophon

Cover design and lay-out:	Tim van de Kamp
Text typefaces:	Freight Text Pro, Mr Eaves XL Sans, Gemeli Mono
Math typefaces:	Latin Modern Math, XITS Math, TeX Gyre Chorus, TeX Gyre Pagella Math
Typesetting software:	LuaLaTeX, biber
Paper:	90 gsm wood-free offset
Printing:	Ipskamp Printing
Publisher:	University of Twente Digital Society Institute (DSI) P.O. Box 217 7500 AE Enschede

The printing of this dissertation was financially supported by the Services and Cybersecurity (SCS) research department of the University of Twente.

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE