


Numéro National de Thèse : -

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
l'École Normale Supérieure de Lyon

École Doctorale N°512
École Doctorale en Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique

Soutenue publiquement le /09/2020, par :
Radu Ţiţiu

New Encryption Schemes and Pseudo-Random Functions with Advanced Properties from Standard Assumptions

---

Devant le jury composé de :

Castagnos Guilhem, Maître de Conférences, Université de Bordeaux	Rapporteur
Steinfeld Ron, Senior Lecturer, Monash University (Australie)	Rapporteur
Phan Duong Hieu, Professeur des Universités, Université de Limoges	Examinateur
Ràfols Carla, Tenure Track Lecturer, Univestitat Pompeu Fabra (Espagne)	Examinatrice
Libert Benoît, Directeur de Recherche, CNRS et ENS de Lyon	Directeur de thèse

New Encryption Schemes and Pseudorandom Functions with Advanced Properties from Standard Assumptions

Radu ŢIŢIU

Supervisor: Benoît LIBERT

June 17, 2020

Abstract

In this thesis, we study the security of advanced cryptographic primitives against adversaries that behave closer to real-life scenarios. Namely, they can adaptively update their strategy during the attack, based on previously obtained information, possible from external sources like corrupted users. We construct Distributed Pseudorandom Functions that still output random-looking values, even when the adversary can adaptively corrupt some servers. Such a system assumes that the secret key is shared among multiple servers that have to combine their partial evaluations in order to obtain a pseudorandom value. We also prove security against adaptive corruptions, in the stronger simulation-based security model, for Inner Product Functional Encryption. Such a public-key scheme encrypts vectors x and can issue multiple secret keys associated to key vectors y. The decryptor learns the partial information <x,y> but nothing else. This primitive can compute statistics (e.g., weighted sums or means) on a database, while keeping each individual input private. We also construct a labeled variant, wherein each database entry is encrypted by a different client, called Multi-Client Functional Encryption. We finally provide a new construction of Non-Interactive Zero-Knowledge proof, which convinces a verifier of the validity of some NP statement without leaking anything else. In addition, an adversary obtaining many simulated proofs for possibly false statements cannot produce a valid proof of its own for a false statement. This primitive is used as a building-block for public-key encryption schemes with advanced security properties.

Contents

Chapter

1

Introduction

1.1 Background

The word "cryptography" is typically associated with the private communication of messages through secret writings and their deciphering, or *encryption* and *decryption* as we call them today. This meaning prevails even though the science of cryptography solves a wide variety of problems besides privacy. This should not come as a surprise since for thousands of years its only purpose was to prevent unintended interceptors from reading and understanding the messages that were being sent. Usually both the sender and the receiver possessed the same secret information, that they had agreed upon prior to the communication, by meeting face to face or through other trusted means. We refer to such encryption schemes, where the same secret information must be used to encrypt and decrypt messages, as *symmetric-key*. Once a secret key is shared between two entities that want to communicate (on the Internet, this is achieved through *public-key* methods), the symmetric-key branch of cryptography is able to solve other problems beside encryption, problems related to authentication and message integrity. This is usually done through the use of *heuristically secure*, but extremly fast algorithms like AES. A more theoretical approach, that provides *provable security*, is to construct fundamental symmetric-key primitives using *Pseudo-Random Functions* (PRFs) [GGM86] as building blocks.

With the development of computer networks from the beginning of the '70s, cryptography had to deal with new problems related to privacy, authentication and data integrity. The *public-key* revolution [DH76] introduced the new ideas of *public-key cryptography* and *digital signatures* which answered questions like: "how to protect the privacy of the data against eavesdroppers, when communicating with people on the other side of the continent, whom we have never spoken to before?" or "how to make sure we are communicating with the intended people and not with someone impersonating them?" and "how do we make sure that the received data was not modified somewhere along the communication chan-

nel?" This new paradigm assumed generating a pair of related keys: a public key, used for encryption, and a private key for decryption. Encrypting using publicly available information removes the need for an a priori sharing of the secret key. The relation between the public and the private key, allows for the secret key to be *mathematically* recovered from the information encapsulated by the public key. But in practice, the task of computing the secret key in this manner is *computationally* intractable, even using the most powerful computers currently available. So the security of public-key cryptography has to rely on the conjectured harness of some computational task, or *computational assumption*.

These powerful ideas heavily influenced the development of the Internet as we know it today. In fact, the key-exchange protocol of Diffie and Hellman [DH76] used for key agreement by complete strangers, and the popular RSA public-key encryption and digital signature [RSA78], are nowadays in constant use.

The Internet boom from the last couple of decades brought about millions of everyday connections from small devices like mobile phones of tablets. The small storage capabilities and restricted computational power of such devices enabled cloud computing services to become very popular among individual internet users and companies alike. The drawback for clients moving their computation to the cloud is that they need to hand over sensitive data in clear to third parties they might not trust. This inherent tension between privacy and the ability of performing useful operations with the data can be relieved for some applications using *advanced* cryptographic primitives. For example, secure Multi-Party Computation (MPC) [Yao86, GMW87] gives an interactive protocol that allows multiple users to compute and learn the output of a function evaluated on their inputs, while keeping each individual input private. Fully Homomorphic Encryption (FHE) [Gen09, BGV11, BV11, GSW13] allows untrusted parties to compute on user's encrypted data, after which the user decrypts the result of the computation. Functional Encryption (FE) [O'N10, BSW11a] is a public-key encryption with advanced functionality, where multiple decryption keys can be issued. Each key is associated to a computation such that the decryptor learns only the result of the computation on the encrypted data, but nothing else. Functional Encryption can be seen as a non-interactive form of a MPC protocol with only two participants, also known as 2-Party-Computation (2PC).

This thesis

One object that we study in this thesis is the fundamental cryptographic primitive of Pseudo-Random Functions (PRFs) [GGM86], which can be efficiently used to get essential symmetric-key constructions such as encryption or message authentication. We study this primitive in the context of threshold cryptography, in which the secret key is shared among many servers such that any sufficiently large subset can correctly evaluate the PRF. This improves the fault-tolerance properties of the system by keeping it functional despite some small number of possible faulty servers. In this setting where the secret key is not stored in one place, but

distributed among multiple entities, it is less likely that an adversary will ever be able to corrupt sufficiently many servers to compromise the whole system. For these reasons, we could use this primitive for long-term encryption of sensitive data.

In the second part, we consider Functional Encryption schemes for the restricted class of linear functions, also called Inner-Product Functional Encryption (IPFE) [ABDCP15], which can be applied to compute weighted sums or means of some encrypted database, while nothing else is leaked about it. When the encrypted data can be supplied by multiple clients, each one encrypting with its own individual secret key, we call such scheme Multi-Client Functional Encryption (MCFE) [CDG+18a].

In a final contribution, we pay attention to public-key encryption schemes providing key-dependent message (KDM) [BRS02] security under adaptive chosen-ciphertext attacks (CCA2). KDM security allows a scheme to encrypt messages that depend on the secret key itself, while still guaranteeing privacy. Beyond its theoretical appeal, such an advanced security is desirable in situations that may appear due to careless key management or when using disk encryption utilities [BHHO08].

For the construction of the KDM-CCA2 scheme, we use non-interactive zero-knowledge (NIZK) [BFM88] proof systems, which enable a prover to produce a proof that some statement is true while hiding everything else. Anyone who runs the verification procedure must be convinced of the validity of the statement, while learning nothing else beyond this fact. This seemingly paradoxical concept has many other cryptographic applications. For instance, it can be used to enforce honest protocol behavior, while providing privacy for the involved users, or it can be used as a building-block to obtain digital signatures.

In this thesis, we provide new constructions for the above-mentioned primitives, with focus on security against adversaries that behave closer to real-life scenarios. Namely, they can *adaptively* update their strategy during the attack, based on previously obtained information, possible from external sources like corrupted users.

1.2 Provable Security

There have been many examples of broken encryption schemes throughout history, so one may wonder if it is possible to build schemes for which we can mathematically *prove* that no attacks are possible, regardless of the computational resources of the adversaries and their attack strategies.

Shannon showed in his seminal paper [Sha48] that, unless the secret key is at least as large as the message to be encrypted, a scheme inevitably leaks information about the message, which an all-powerful adversary could learn. So encryption with perfect secrecy is not really practical as it requires prohibitively large keys. This means that we can only hope for practical cryptography by mak-

ing the assumption that adversaries are not all-powerful, but are computationally bounded. Even though a scheme with short secrets leaks information, the adversary might not have the necessary computational power to exploit this. For this reason, we consider only *efficient* adversaries that have bounded polynomial resources (e.g., their running time is bounded by some polynomial in the input length). This might appear like a powerful restriction, but seems enough to capture all real world adversaries.

As we have already mentioned, a secure scheme is one for which the problem of exploiting the leaked information should be computationally intractable for any efficient adversary. Therefore if we want to prove security, it is necessary to at least assume hard computational problems, in the class of problems for which we can efficiently verify solutions (also known as NP), that cannot be solved in a polynomial number of steps. Unfortunately, proving the existence of such an intractable problem would settle one of the most important open problems in computer science, namely P versus NP. Hence, as long as this problem remains open, we cannot hope to have unconditional security proofs, but they must depend on the conjectured hardness of some computational problem (computational assumption). The assumption that hard problems exist is supported by both our decades-long experience with computers and decades of complexity theory research.

Security Proofs

In order to give a security proof, we should first define what it means for a scheme to be secure. The security intuitions for a scheme must be formalized into a precise definition. Very often, such a definition is given as an interactive game between the adversary and the challenger. The scheme is then considered broken if the adversary wins the game.

A security proof for a cryptographic scheme is a theorem which formally shows that it is impossible for an efficient adversary to win the security game (thus the scheme cannot be broken) as long as some computational assumption holds. Such a theorem is usually proved by assuming the existence of an efficient adversary that is able to break the cryptosystem. The proof then proceeds to construct another efficient algorithm, that runs this adversary as a subroutine, and that is able to solve the underlying computational assumption, arriving at a contradiction. The efficient algorithm, that uses the adversary as a subroutine, and breaks the computational assumption is known as the *reduction*.

Standard Assumptions

In cryptography, the term *standard assumptions* refers to a set of computational problems that have been used for constructing many different primitives and that are believed to be intractable by quasi-polynomial time algorithms. The confidence that we have in these assumptions comes from decades of research done by mathematicians, cryptanalysts and computer scientists that worked on these

problems but failed to solve them. For example, maybe one of the most popular assumptions, is the Discrete Logarithm (DL) assumption which is used on the Internet on a daily basis. Given a cyclic group G of prime order p and a generator g, the DL assumption in group G, states that it is computationally hard to compute $x \in \mathbb{Z}_p$ from g^x. Obviously the difficulty of the problem depends on the group G. The confidence in an assumption like this only grows with each passing day that remains unbroken.

1.3 Pseudo-Random Functions

A Pseudo-Random Function (PRF) family [GGM86] is a family of keyed functions $\mathcal{F} := \{F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}$, that can be efficiently evaluated on input a secret key $\mathbf{s} \in \mathcal{K}$, and a value $x \in \mathcal{X}$ from the domain of the function. Security guarantees that no efficient adversary is able to distinguish between the following two scenarios: if it is given oracle access to a function $F_{\mathbf{s}}$, for a uniformly random key $\mathbf{s} \leftarrow\!\!\$\ \mathcal{K}$, or oracle access to a truly random function $f \leftarrow\!\!\$\ \mathcal{Y}^{\mathcal{X}}$, sampled uniformly random from the set of all functions that map $\mathcal{X}$ to $\mathcal{Y}$.

PRFs are fundamental building blocks in cryptography, notably in symmetric-key schemes, where they can be used for solving essential tasks like secret-key encryption, authentication or message integrity.

Goldreich, Goldwasser and Micali [GGM86] showed how to build a PRF from any length-doubling pseudo-random generator (PRG). In turn, PRGs are known [HILL99] to exist under the sole assumption that one-way functions exist. However, this generic way of constructing them is particulary inefficient. Much more efficient constructions can be obtained by directly relying on more standard number theoretic assumptions like the Decision Diffie-Hellman assumption [NR97] and related variants [DY05, LW09, BMR10, CM04] or the hardness of factoring [NR97, NRR00]. Constructions from lattice-based assumptions like, Learning With Errors (LWE) [BPR12, BLMR13, BP14] are also possible, but most of them are rather inefficient. Very often this is due to to the need for a super-polynomial modulus.

Distributed PRFs

Distributed Pseudo-Random Functions (DPRFs) [NPR00] allow distributing between N servers the computation of a function that outputs values that are indistinguishable from random as long as the secret key is not revealed. Moreover, only authorized subsets of the servers are able to compute the correct value. For instance, in the case where the authorized sets consists of all the sets of cardinality above a threshold t, a user who wants to compute the value $F_{SK}(x)$ sends the input x to a subset of t servers which reply with the corresponding partial evaluations on the input x. The user combines all the partial evaluations to recover the evaluation $F_{SK}(x)$. An attacker should not be able to distinguish the value $F_{SK}(x)$ from a uniformly random value, even if it has full control over a maximum number of $t-1$ servers.

As in the general case of threshold cryptography [DF89], the main advantage of having mutiple servers keeping the secret shares and doing the corresponding partial evaluations is that single points of failure are removed, thus keeping the centralized equivalent of the secret key safer and making the whole system more dependable. This means that the whole system continues to function despide the malfunctioning of a subset of the servers.

Distributed pseudo-random functions were initially suggested by Micali and Sidney [MS95] and received a lot of attention since then [NPR00, NR97, Nie02, Dod03, DYY06]. They are motivated by the construction of distributed symmetric encryption schemes, distributed key distribution centers [NPR00], or distributed coin tossing and asynchronous byzantine agreement protocols [CKS00]. They also provide a distributed source of random coins that allows removing interaction from threshold decryption mechanisms, such as the one of Canetti and Goldwasser [CG99].

Previous Work on DPRFs

All the previous DPRF constructions can be divided into two broad categories: interactive and non-interactive. *Interactive* [Nie02, Dod03, DYY06] means that, in order to compute the function, servers need to interact not only with the user, but also between themselves. It is desirable to a have *non-interactive* schemes [MS95, NPR00, BLMR13, BGG$^+$18], where the servers only have a one round interaction with the user. Some of the drawbacks of these first constructions were that they worked only for small values of the threshold t in relation to the number of servers N [MS95], or that the security proof was not solely based on standard assumptions, but used the Random Oracle Model (ROM) [NPR00]. The ROM [BR93] refers to a heuristic security model in which the hash functions used in the scheme are modeled as truly random functions. Moreover, the adversary does not compute these functions itself, but rather invokes an oracle (controlled by the reduction in the security proof) whenever it wants to obtain a value of the hash function. Security proofs that do not use this heuristic are preferred, as there are examples of secure encryption schemes in the random oracle model, for which implementations of the random oracles yield insecure schemes [CGH98].

In 2013, Boneh et al [BLMR13] gave a generic construction of non-interactive DPRFs from Key-Homomorphic PRFs, that works for any threshold values of t, including the honest majority case $t = (n-1)/2$. Their scheme is secure under the standard Learning With Errors assumption, without random oracles. The same generic construction can be instantiated using the KH-PRF from [BP14, Kim20], with more efficient parameters.

Generic DPRF constructions are only proven to be secure in the weaker *static* corruption model. This means that an attacker is restricted to declaring the list of corrupted servers before the attack starts. A more natural and realistic model would be the *adaptive* one, in which the attacker can choose which servers to corrupt in an adaptive manner, based on information received throughout the attack.

It is easy to prove that the adaptive corruption model makes the adversary strictly stronger as we can build a scheme which is insecure under adaptive corruptions, but secure under static corruptions (as we show in Chapter 3).

Contribution 1

The results from [LST18], presented in Chapter 3, give the first non-interactive construction of a Distributed Pseudo-Random Functions (DPRF) family that simultaneously achieves security against *adaptive* corruptions. Security is based on the Learning With Errors assumption, without relying on random oracles. The efficiency of our scheme could be improved, as we need a super-polynomial LWE modulus. Lattice-based PRFs [BPR12, BLMR13, BP14], that use the rounding technique of [BPR12], suffer from this, including our construction.

1.4 Functional Encryption

Functional Encryption (FE) is a modern cryptographic primitive that offers fine-grained access to the encrypted data. It generalizes traditional public-key encryption, where the decryptor either recovers the whole message or nothing.

An FE scheme is usually defined over a class of functionalities $\mathscr{F}$ that take input from the message space of the scheme. The holder of the master secret key is able to issue multiple functional decryption keys sk_f, each associated with some function $f \in \mathscr{F}$. A user that holds such a decryption key sk_f and a ciphertext corresponding to the message $\boldsymbol{x}$ can recover $f(\boldsymbol{x})$. Security guarantees that nothing else is leaked about the message $\boldsymbol{x}$, besides the evaluation $f(\boldsymbol{x})$. Even when multiple users collude, they should still not learn anything else about the message, beyond what each individual key reveals.

The study of the abstract notion of Functional Encryption (FE) was initiated by Boneh, Sahai and Waters [BSW11a] and O'Neil [O'N10]. Many constructions that fit this generalization, like Identity-Based Encryption (IBE) [Coc01, BF01, GPV08, ABB10], Attribute-Based Encryption (ABE) [SW05, GPSW06, LOS+10, GVW13] or Predicate-Encryption (PE) [LOS+10, KSW08, GVW15a], have been extensively studied before and after the general formulation of FE.

Security for FE

Intensive efforts have been put into formalizing the *correct* security definition that captures the security intuition for this primitive. In [BSW11a] and [O'N10], it was shown that the usual *indistinguishability-based* (IND) security notion is not suitable for certain functionalities. More precisely, they gave a trivially insecure FE construction that could be proven secure with respect to the IND security requirement. Informally, the IND-security requires that no efficient adversary that has

oracle access to the key generation algorithm should be able to distinguish between encryptions of two messages $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. This covers the scenario when multiple private key holders collude in order to learn information about the encrypted message. In the same work, they initiated the study of *simulation-based* (SIM) security for FE, a stronger model for this primitive, compared to IND, which asks that the view of the adversary can be simulated by a simulator that is given access to pairs $(f, f(\boldsymbol{x}^\star))$ of functionalities together with their evaluation on the challenge message $\boldsymbol{x}^\star$.

Depending on whether the adversary is allowed to make key queries after the challenge, we consider *adaptive* (AD) and *non-adaptive* (NA) security for both IND and SIM definitions. O'Neill [O'N10] showed in the non-adaptive case, the two notions of security NA-IND and NA-SIM are equivalent for the class of functionalities for which computing preimages is easy.

It was shown by Boneh, Sahai and Waters [BSW11a] that AD-SIM-security is impossible to achieve for many challenge messages for the Identity Based Encryption (IBE) functionality. The same argument easily extends to Inner-Product Functional Encryption (IPFE). Another impossibility result was given by Agrawal et al. [AGVW13] where they proved that we cannot hope to construct an FE scheme for general circuits that achieves NA-SIM security for one challenge message.

At the other side of the spectrum, in terms of positive results, Gorbunov, Vaikuntanathan and Wee [GVW12] gave a construction for the class of general circuits that is AD-SIM secure for one message under *bounded collusion*. This means that there is an a priori bound on the number of key queries made by the adversary. Another construction for general circuits appeared in [GKP+13] achieving NA-SIM security when the adversary is restricted to only one key query. Fully secure (AD-IND) FE for general circuits can be constructed using the powerful machinery of indistinguishability obfuscation and multi-linear maps [GGH+13, GGHZ14, Wat15]. For the time being, these tools are not known to be instantiable under well-studied hardness assumptions.

Inner-Product Functional Encryption

When we restrict the class of functionalities to linear functions, that evaluate inner-products of vectors over some ring R ($R \in \{\mathbb{Z}, \mathbb{Z}_p\}$), we call the resulting Inner-Product Functional Encryption (IPFE). Concretely, IPFE encrypts vectors $\boldsymbol{x} \in R^\ell$ using the public key. Using the master secret key, it can also issue multiple secret keys associated with key vectors $\boldsymbol{y} \in R^\ell$. The decryptor learns the partial information $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \in R$ but nothing else.

Previous Work on IPFE

Instead of aiming at building FE for general circuits from standard assumptions, another approach was adopted by Abdalla, Bourse, De Caro and Pointcheval in [ABDCP15]. They considered building efficient FE, starting with smaller classes of

functionalities instead of aiming for general circuits, with security based on standard assumptions. In particular, for the restricted class of linear functions (inner products) they managed to get more practical schemes under well studied hardness assumptions like the Decisional Diffie-Hellman (DDH) and Learning With Errors (LWE). Their constructions are only proven to be *selectively secure* in the IND model. This means that the adversary has to announce the challenge messages before it even sees the public key of the scheme. This result was improved by Agrawal, Libert and Stehlé [ALS16] by constructing adaptive AD-IND functional encryption for the same class of linear functions, under DDH , LWE and from Paillier's Decision Composite Residuosity (DCR) assumption [Pai99]. Going even further, Wee [Wee17] proved that the DDH based FE scheme from [ALS16] satisfies a SIM security notion called *semi-adaptive simulation* security. In this setting, the adversary is not allowed to make any key queries before the challenge.

Contribution 2

For our second contribution, we focus on our results from [ALMT20]. In Chapter 5, we show the IPFE schemes of [ALS16], based on DDH, Paillier's DCR [Pai99] and LWE, that evaluate inner products over the integers, are in fact AD-SIM secure. This also improves the result from [Wee17], where the DDH-based scheme was proved to be semi-adaptive simulation secure. To prove these results, we have to increase the size of the secret keys of the original DCR-based scheme. We also need to enlarge the modulus q, of the LWE-based scheme of [ALS16], by an exponentially large factor. In the DDH case, we can prove our results without any modifications in the original scheme.

Since the impossibility result of [BSW11a] can be adapted to the inner-product functionality, it excludes the possibility of achieving AD-SIM security when the adversary is allowed to issue multiple challenge messages. However, none of the impossibility results from [BSW11a, O'N10, AGVW13] applies to our case of a single ciphertext challenge and unbounded number of key queries. Therefore, we prove that the DDH, DCR and LWE-based IPFE schemes from [ALS16] achieve the strongest security notion that we can hope for among the IND and SIM based definitions, namely it provides AD-SIM security for single challenge ciphertexts for unbounded [1] number of key queries.

Multi-Client Functional Encryption

Multi-Input Functional Encryption (MIFE) [GGG+14] allows the computation of functions over several inputs that can be encrypted independently using the same

[1] The number of key queries is actually bounded by the functionality itself, but this is intrinsic to FE. For instance, IPFE security becomes void after ℓ linearly independent key queries, where ℓ is the dimension of the message vectors

key. When each input is encrypted by a different client or party, under its own private key, we refer to such schemes as multi-client (MC) [GGG+14, CDG+18a]. In this scenario it is natural to assume that corrupt clients may collude in order to learn more information about the remaining inputs. On the other hand, in a MIFE scheme there is no security against corruption, since all parties encrypt using the same key.

Labels

Both in the multi-input and multi-client settings, the inherent information leakage could be a problem in applications, especially when many functional decryption keys are given out. To illustrate this point, consider the simple 2-client case. Anyone who possesses a functional decryption key sk_f for the two-input function f and the encryptions of x_1, z_1 and x_2, z_2, computed by the first and the second client, respectively, can learn any $f(x_i, z_j)$ of the four combinations. Notice that, for ℓ clients, the information leakage induced by one functional decryption key on two ciphertext vectors grows exponentially with ℓ, the number of clients.

To limit this "mix-and-match" leakage, the idea of using *labels* was introduced in [GGG+14]. This means that each party encrypts its message using a label t as an additional parameter (for example a timestamp) and any given functional decryption key can only operate over ciphertext vectors encrypted over the same label. Chotard et al. proposed in [CDG+18a] the first efficient MCFE scheme under standard assumptions, that also supports labels. Their construction works for the restricted class of linear functions. Notice that such schemes with labels can be obtained from MCFEs for general functionalities. Unfortunately, such constructions are terribly inefficient and are based on exotic assumptions like Indistinguishability Obfuscation [GGG+14].

Previous Work on MCFE

The first standard model MCFE with labels from [CDG+18a] is secure under the DDH assumption, but in the Random Oracle Model, and works for the restricted class of linear functions. In their security model (Definition 4.2), each client is assumed to encrypt only one message per label. Moreover, the adversary is assumed to know all the ciphertexts encrypted under the same label, corresponding to each client. This means that the security is not guaranteed when the adversary is allowed to obtain ciphertexts for a proper subset of the clients for a particular label. We will discuss more about these limitations in Section 4.1.

In [CDG+18b], the same authors enhance the security of the previously mentioned [CDG+18a] schemes by giving two transformations that remove the need for these seemingly artificial conditions. Unfortunately their techniques are not generic and only work for their initial scheme. They deal with partial ciphertexts by using a Secret Sharing Layer (SSL), which is used to encapsulate the ciphertexts such that the recovery is possible only when the contributions of all the clients are

available. In addition, by using an IPFE [ALS16] encryption layer, similarly to the IPFE to MIFE compiler of [ACF+18], they removed the clients' restriction of one ciphertext per label.

A similar result in [ABKW19] shows how to generically upgrade MCFE schemes so as to prove security in the case of incomplete ciphertexts. For schemes that do not support labels, their compiler can be instantiated with an IND-CPA symmetric encryption scheme and is similar to the one in [AGRW17]. In case of labeled MCFE, the use of symmetric encryption is not sufficient, as the compiler's security is proven in the Random Oracle Model. Another result from [ABKW19] shows that the standard model MIFE schemes of [ACF+18] remain secure against adaptive corruptions. Thus we may refer to them as MCFE schemes.

Finally, in the concurrent work of [ABG19] the authors give a general compiler that takes any single-input IPFE (that satisfies some mild conditions) and gives an MCFE with labels with security under adaptive corruptions. Analogously to the transformation from [ACF+18], by instantiating the compiler with the IPFE schemes from [ALS16], they obtained labeled MCFEs under the LWE/DDH/DCR assumptions. They also upgrade the proof of the compiler in [ABKW19] to work in in standard model, thus providing security when incomplete ciphertext vectors are allowed.

Contribution 3

Our contributions of [LT19] are presented in Chapter 4 of this thesis. The construction from Section 4.3 gives the first MCFE scheme for linear functions, secure under adaptive corruptions, that supports labels. The security is guaranteed by the LWE assumption and does not require random oracles.

Scheme (MCFE)	Standard Assumptions	Labels	Adaptive Corruptions	Stand. Model	Partial Ciphertexts	Multiple ct per label
[Abd.+18] Crypto18	DDH/LWE/DCR ✔	✘	✘	✔	✘	✘
[Chot.+18] AC18	DDH ✔	✔	✔	✘	✘	✘
[ABKW19] PKC19	DDH ✔	✔	✔	✘	✔	✘
[ABG19] AC19	DDH/LWE/DCR ✔	✔	✔	✔	✔	✔
This work AC19	LWE ✔	✔	✔	✔	✔	✘

Figure 1.1: Our Contribution

Another contribution is detailed in Section 4.4 where a compiler upgrades the basic security of our MCFE scheme so that it supports partial ciphertexts. The security proof is in the standard model and makes use of the *adaptive multi-instance PRFs*, which can be proved secure by leveraging the pseudo-randomness of any PRF. To get better tightness factors in the security proof, we use the particular LWE-based PRF from section 3.4.2.

1.5 Non-Interactive Zero-Knowledge Argument Systems

Non-Interactive Zero-Knowledge (NIZK) proof systems are protocols in which a prover can convince an efficient verifier that some NP statement is true, while not leaking anything else. The protocol requires no interactive conversation between the prover and the verifier, as the prover computes a proof that should convince the verifier of the validity of the statement. They were shown to exist [BFM88] in the *Common Reference String* (CRS) model, where the prover and the verifier share a string of randomness. In the plain model, NIZKs are only possible for languages in the complexity class of bounded probabilistic polynomial time (BPP) [Ore87], so the shared CRS is in fact necessary.

One popular approach to construct non-interactive zero-knowledge (NIZK) proof systems is to apply the Fiat-Shamir (FS) transform [FS86] to 3-move interactive, honest-verifier zero-knowledge protocols, also known as Σ-protocols. Concretely, the interaction from the Σ-protocol is removed by replacing the verifier's random challenge by a value computed by a deterministic hash function, when applied to the transcript up to that point. Bellare and Rogaway [BR93] showed that the non-interactive scheme is sound, provided the hash function is modeled as a random oracle. However, the Fiat-Shamir heuristic may be insecure in the standard model, when the hash function is no longer the ideal random function [GTK03]. Negative results like [GTK03] did not rule out the soundness of Fiat-Shamir-based NIZK proofs when a specific hash function is applied to specific Σ-protocols.

Until recently, it was not known how to instantiate the Fiat-Shamir paradigm in the standard model. This changed with the recent works of Canetti *et al.* [CLW19] and Peikert and Shienhian [PS19], who gave the first *Correlation-Intractable* (CI) hash functions family, secure under standard lattice assumptions. They showed that CI for searchable relations is sufficient to soundly instantiate the FS paradigm for a sufficiently large class of Σ-protocols such that NIZK for all NP relations is possible under standard lattice assumptions.

Previous Work on Multi-Proof NIZKs

Sahai showed [Sah99] that, when the double-encryption paradigm of Naor and Yung [NY90] is instantiated using a *simulation-sound* NIZK, we obtain a public-

key encryption scheme that is secure against chosen-ciphertext attacks (CCA2). To this end, he showed how to transform any ordinary NIZK to obtain simulation-soundness. Intuitively, simulation-soundness prevents an adversary from producing a valid proof for a false statement, even if it is in possession of one other proof for a statement of his choosing, obtained from other sources.

This result was later improved for the multi-proof case in [SCO+01], which gives a general transformation from ordinary NIZKs to *unbounded simulation-sound* NIZKs. Intuitively, this prevents the adversary from producing a valid proof corresponding to a false statement, even after seeing polynomially many other proofs of its choosing.

Again, in the multi-proof setting, the results from [FLS99] give a general transformation from any ordinary NIZK to multi-theorem zero-knowledge proof systems. This means that the zero-knowledge property of the scheme is preserved even when polynomially many proofs are given to the adversary.

Contribution 4

In the last part of the thesis we present some of our results from [LNPT19], which are concerned with more efficient instantiations of the Naor-Yung paradigm, for specific languages.

Our first result (Section 6.3.2) gives a generic construction of a multi-theorem NIZK argument [2], directly from trapdoor Σ-protocols, assuming CI hash functions and a primitive called "lossy encryption with efficient opening" [BHY09]. We use the CI hash functions to make the proof system non-interactive, while preserving soundness. The fact that lossy ciphertexts can be easily equivocated [3] is crucially exploited by the NIZK simulator and it is the ingredient that allows us to prove zero-knowledge when the adversary has access to polynomially many proofs. As already mentioned, it is also possible to obtain multi-theorem NIZK through the FLS transformation [FLS99]. Our results give an alternative way of doing that, directly from Σ-protocols. As a consequence of our techniques, we can obtain multi-theorem statistical NIZKs for all NP languages under the standard LWE assumption (see Section 6.3.4), which has an advantage over the same result obtained by combining the results from [FLS99] and [PS19]. Namely, in our case we obtain a uniformly distributed CRS while the CRS obtained by the FLS transformation is only pseudo-random.

Our second contribution, given in Section 6.4, shows how to upgrade the previously mentioned multi-theorem construction (Section 6.3.2) to obtain unbounded simulation-soundness as well. In order to do this, we need to also use one-time signatures as a building block. Unfortunately, standard lossy encryption with efficient opening is not enough to prove simulation-soundness. For this purpose, we generalize the definition of R-lossy encryption [BSW11b] and prove (in Section

[2] An *argument* is a proof with *computational* soundness

[3] A ciphertext is not committed to any particular message. In fact, it can be explained as the encryption of any message from the set of all possible messages

6.2) that a variant of Regev's primal scheme satisfies all the requirements. This results in a generic compiler that takes any trapdoor Σ-protocol and transforms it into a multi-theorem NIZK argument that satisfies the unbounded simulation-soundness property. Again, our results can be seen as an alternative to [SCO+01] of obtaining unbounded simulation-sound NIZKs. Both of our constructions provide statistical zero-knowledge if the underlying Σ-protocol satisfies statistical special zero-knowledge. Moreover, in both cases, the language of the underlying trapdoor Σ-protocol is exactly the same as that of the resulting NIZK argument.

Finally, in Section 6.4.5 we give a construction for a trapdoor Σ-protocol, that proves two ciphertexts of the LWE-based KDM-CPA secure scheme of [ACPS09], encrypt the same message. Together with the generic compiler from Section 6.4, it allows us to apply the Naor-Yung transform to the ACPS scheme. This yields the most efficient public-key scheme, key-dependent message (KDM) secure under chosen-ciphertext attacks (CCA2), under the LWE assumption. This result is discussed in Section 6.4.4.

Chapter

2

Preliminaries

In this chapter, we recall some basic notation, definitions and some results from the literature that we will use in this thesis.

2.1 Notation

For any $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers with addition and multiplication modulo q. For $2 \leq p < q$ and $x \in \mathbb{Z}_q$, we define $\lfloor x \rfloor_p := \lfloor (p/q) \cdot x \rceil \in \mathbb{Z}_p$. This notation is extended componentwise to vectors over $\mathbb{Z}_p$. If $\mathbf{x}$ is a vector over $\mathbb{R}$, then $\|\mathbf{x}\|$ denotes its Euclidean norm $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots x_n^2}$. If $\mathbf{M}$ is a matrix over $\mathbb{R}$, then $\|\mathbf{M}\| := \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{M}\mathbf{x}\|}{\|\mathbf{x}\|}$ denotes its induced norm. We can also define the infinity norm $\|\mathbf{x}\|_\infty := \max_i |x_i|$ and its induced matrix norm $\|\mathbf{M}\|_\infty := \sup_{\mathbf{x} \neq 0} \frac{\|\mathbf{M}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty}$. We let $\sigma_n(\mathbf{M})$ denote the least singular value of $\mathbf{M}$, where n is the rank of $\mathbf{M}$.

For a finite set S, we let $U(S)$ denote the uniform distribution over S. If X is a random variable over a countable domain, the min-entropy of X is defined as $H_\infty(X) = \min_x(-\log_2 \Pr[X = x])$. If X and Y are distributions over the same domain, then $\Delta(X, Y)$ denotes their statistical distance.

Let $\sqrt{\Sigma} \in \mathbb{R}^{n \times n}$ be a non-degenerate matrix, and $\mathbf{c} \in \mathbb{R}^n$. We define the Gaussian function on $\mathbb{R}^n$ by $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp(-\pi(\mathbf{x}-\mathbf{c})^\top \Sigma^{-1}(\mathbf{x}-\mathbf{c}))$, for $\Sigma := \sqrt{\Sigma} \cdot \sqrt{\Sigma}^\top$. If $\sqrt{\Sigma} = \sigma \cdot \mathbf{I}_n$ we denote it by $\rho_{\sigma,\mathbf{c}}$. In addition if $\mathbf{c} = \mathbf{0}$ as well we denote it by ρ_σ.
By PPT, we refer to a probabilistic polynomial time algorithm.

2.2 Lattices

Given n linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^n$, the lattice generated by them is defined as the set of all the integer linear combinations:

$$\Lambda(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) := \mathbb{Z} \cdot \mathbf{b}_1 + \mathbb{Z} \cdot \mathbf{b}_2 + \cdots + \mathbb{Z} \cdot \mathbf{b}_n,$$

or we can write this in a compact manner as $\Lambda(\mathbf{B}) = \mathbf{B} \cdot \mathbb{Z}^n$, where the matrix $\mathbf{B} = [\mathbf{b}_1 | \mathbf{b}_2 | \cdots | \mathbf{b}_n] \in \mathbb{R}^{n \times n}$. We call the matrix $\mathbf{B}$ a *basis* of the lattice. A lattice has an infinite number of bases. Any two bases are related through a unimodular matrix.

For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define the modular lattices

$$\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \bmod q\}, \qquad \Lambda(\mathbf{A}) = \mathbf{A}^{\top} \cdot \mathbb{Z}^n + q\mathbb{Z}^m.$$

For an n dimensional lattice $\Lambda \subset \mathbb{R}^n$ and for any lattice vector $\mathbf{x} \in \Lambda$, the discrete Gaussian is defined $\rho_{\Lambda,\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x})}{\rho_{\sqrt{\Sigma},\mathbf{c}}(\Lambda)}$. For an n-dimensional lattice Λ, we define the *smoothing parameter* $\eta_\varepsilon(\Lambda)$ as the smallest $r > 0$ such that $\rho_{1/r}(\widehat{\Lambda} \setminus \mathbf{0}) \leq \varepsilon$ with $\widehat{\Lambda} := \{\mathbf{y} \in \mathbb{R}^n : \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z} \text{ for all } \mathbf{x} \in \Lambda\}$ denoting the dual of Λ, for any $\varepsilon \in (0,1)$. The notation $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ means that $1 \geq \eta_\epsilon(\sqrt{\Sigma}^{-1} \cdot \Lambda)$. We also define $\lambda_1^{\infty}(\Lambda) = \min(\|\mathbf{x}\|_\infty : \mathbf{x} \in \Lambda \setminus \mathbf{0})$.

Below we recall some useful lattice-related lemmas.

Lemma 2.1 ([MR07, Lemma 4.4])**.** *For an n-dimensional lattice $\Lambda \in \mathbb{R}^n$ any vectors $\mathbf{c}, \mathbf{u} \in \mathbb{R}^n$, $\epsilon \in (0,1)$ and a positive definite matrix Σ such that $\sqrt{\Sigma} \geq \eta_\epsilon(\Lambda)$ we have:*

$$\Pr_{\mathbf{x} \hookleftarrow D_{\Lambda+\mathbf{u},\sqrt{\Sigma},\mathbf{c}}} \left[\|\mathbf{x} - \mathbf{c}\| > s_1(\sqrt{\Sigma}) \cdot \sqrt{n} \right] \leq 2^{-n} \cdot \frac{1+\epsilon}{1-\epsilon}$$

Lemma 2.2 ([GPV08, Theorem 4.1])**.** *There is a PPT algorithm that, given a basis $\mathbf{B}$ of an n-dimensional $\Lambda = \Lambda(\mathbf{B})$, a parameter $s > \|\tilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log n})$, and a center $\mathbf{c} \in \mathbb{R}^n$, outputs a sample from a distribution statistically close to $D_{\Lambda,s,\mathbf{c}}$.*

Lemma 2.3 ([GPV08, Lemma 5.3])**.** *If $m \geq 2n \cdot \log q$ and $q \geq 2$ is a prime, then*

$$\Pr_{\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})} \left[\lambda_1^{\infty}(\Lambda(\mathbf{A})) \geq q/4 \right] \geq 1 - 2^{-\Omega(n)}.$$

Corollary 2.3.1. *For $m \geq 2n \log q$, $q > p > 4$ and $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$ the function that maps $\mathbf{x} \in \mathbb{Z}_q^n$ to $\lfloor \mathbf{A}^{\top} \cdot \mathbf{x} \rceil_p$ is injective with overwhelming probability.*

Proof. Assume we have $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^n$ such that they map to the same value. This is equivalent to $\mathbf{A}^{\top}(\mathbf{x} - \mathbf{y}) \in \left(-q/p, q/p\right)^m \cap \mathbb{Z}^m$. For $\mathbf{x} \neq \mathbf{y}$ this gives a point from the lattice $\Lambda(\mathbf{A})$ of infinity norm strictly less than q/p, but this contradicts the lemma. □

Lemma 2.4 (Adapted from [GPV08, Lemma 5.3])**.** *Let $q \geq 2$ be prime and let integers $m > n \geq 1$. Let a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. Then, we have the inequality:*

$$\Pr_{\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})} [\eta_{2^{-m}}\left(\Lambda^{\perp}(\mathbf{A})\right) \leq O(\sqrt{m}) q^{n/m}] \geq 1 - 2^{-m}.$$

Lemma 2.5 (Adapted from [MR07, Lemma 4.4])**.** *For any n-dimensional lattice Λ, $\mathbf{x}', \mathbf{c} \in \mathbb{R}^n$ and symmetric positive definite $\Sigma \in \mathbb{R}^{n\times n}$ satisfying $\sigma_n(\sqrt{\Sigma}) \geq \eta_{2^{-n}}(\Lambda)$, we have*

$$\rho_{\Sigma,\mathbf{c}}(\Lambda + \mathbf{x}') \in [1-2^{-n}, 1+2^{-n}] \cdot \det(\Sigma)^{1/2}/\det(\Lambda).$$

Lemma 2.6. *Let $\varepsilon \in (0,1)$, $c \in \mathbb{R}$ and $\sigma > 0$ such that $\sigma \geq \sqrt{\ln 2(1+1/\varepsilon)/\pi}$. Then*

$$H_\infty(D_{\mathbb{Z},\sigma,c}) \geq \log(\sigma) - \log\left(1 + \frac{2\varepsilon}{1-\varepsilon}\right)$$

Proof. From [MR07, Lemma 3.3] we know that $\eta_\varepsilon(\mathbb{Z}) \leq \sqrt{\ln 2(1+1/\varepsilon)/\pi}$. So $\sigma \geq \eta_\varepsilon(\mathbb{Z})$. By [MP12, Lemma 2.5], this implies that $\frac{1-\varepsilon}{1+\varepsilon} \cdot \rho_\sigma(\mathbb{Z}) \leq \rho_{\sigma,c}(\mathbb{Z})$, which translates into

$$H_\infty(D_{\mathbb{Z},\sigma,c}) \geq H_\infty(D_{\mathbb{Z},\sigma}) - \log\left(\frac{1+\varepsilon}{1-\varepsilon}\right)$$

By Poisson's summation formula, we have $\rho_\sigma(\mathbb{Z}) \geq \sigma$, so $H_\infty(D_{\mathbb{Z},\sigma}) \geq \log\sigma$.

Note that for $\sigma = \Omega(\sqrt{n})$, we get $H_\infty(D_{\mathbb{Z},\sigma,c}) \geq \log(\sigma) - 2^{-n}$. □ □

Lemma 2.7 ([Lyu12, Th. 4.6])**.** *Let V be a subset of $\mathbb{Z}^m$ in which all elements have norms less than T, let σ be a real number such that $\sigma = \omega(T\sqrt{\log m})$, and $h : V \to \mathbb{R}$ be a probability distribution. Then, there exists a real number M such that the distribution of the following algorithm $\mathscr{A}$:*

1: $\mathbf{v} \hookleftarrow h$

2: $\mathbf{z} \hookleftarrow D_{\mathbb{Z}^m,\sigma,\mathbf{v}}$

3: output $(\mathbf{z}, \mathbf{v})$ *with probability* $\min\left(\frac{D_{\mathbb{Z}^m,\sigma}(\mathbf{z})}{MD_{\mathbb{Z}^m,\sigma,\mathbf{v}}(\mathbf{z})}, 1\right)$

is within statistical distance $\frac{2^{-\omega(\log m)}}{M}$ from the distribution of the following algorithm $\mathscr{F}$:

1: $\mathbf{v} \hookleftarrow h$

2: $\mathbf{z} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$

3: output $(\mathbf{z}, \mathbf{v})$ *with probability* $1/M$.

Moreover, the probability that $\mathscr{A}$ outputs something is at least $\frac{1-2^{-\omega(\log m)}}{M}$. More concretely, if $\sigma = \alpha T$ for any positive α, then $M = e^{12/\alpha + 1/(2\alpha^2)}$, the output of $\mathscr{A}$ is within statistical distance $2^{-100}/M$ of the output of $\mathscr{F}$, and the probability that $\mathscr{A}$ outputs something is at least $\frac{1-2^{-100}}{M}$.

Micciancio and Peikert [MP12] described a trapdoor mechanism for LWE. Their technique uses a "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$, with $w = n\log q$, for which anyone can publicly sample short vectors $\mathbf{x} \in \mathbb{Z}^w$ such that $\mathbf{G}\cdot\mathbf{x} = \mathbf{0}$.

Lemma 2.8 ([MP12, Section 5]). *Let $m = 2n\lceil \log q \rceil + O(\lambda)$. There exists a* PPT *algorithm* $\mathsf{TrapGen}(1^m, 1^n)$ *that outputs a statistically uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, together with a trapdoor $\mathbf{T_A} \in \mathbb{Z}^{m\times m}$ for $\Lambda^{\perp}(\mathbf{A})$, such that $\max_j \|\tilde{\mathbf{t}}_j\| \leq O(\sqrt{n\log q})$, where $\tilde{\mathbf{t}}_j$ are the corresponding Gram-Schmidt vectors.*

It is known [MP12] that, for any $\mathbf{u} \in \mathbb{Z}_q^n$, a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ allows sampling from $D_{\Lambda^{\mathbf{u}}(\mathbf{A}), s\cdot\omega\left(\sqrt{\log m}\right)}$ for $s = O(\sqrt{n\log q})$. Since

$$\eta_{2^{-m}}\left(\Lambda^{\perp}(\mathbf{A})\right) \leq \max_j \|\tilde{\mathbf{t}}_j\| \cdot \omega(\sqrt{\log m}) \leq s\cdot\omega(\sqrt{\log m})$$

for large enough $s = O(\sqrt{n\log q})$, the magnitude of a vector $\mathbf{x}$ sampled from the discrete Gaussian $D_{\Lambda^{\mathbf{u}}(\mathbf{A}), s\cdot\omega\left(\sqrt{\log m}\right)}$, is bounded by $\|\mathbf{x}\| \leq s\sqrt{m}\cdot\omega(\sqrt{\log m})$.

Remark 1. For $m \geq 3n\log q$, we can thus sample a statistically uniform matrix $\mathbf{A}$ from $\mathbb{Z}_q^{n\times m}$ together with a trapdoor, which allows finding small solutions of $\mathbf{A}\cdot\mathbf{x} = \mathbf{u} \bmod q$, with $\|\mathbf{x}\| \leq s\sqrt{m}\cdot\omega(\sqrt{\log m}) = O(\sqrt{mn\log q})\cdot\omega(\sqrt{\log m})$.

2.3 Some Useful Lemmas

Lemma 2.9 ([GKPV10, Lemma 3]). *Let $\boldsymbol{y} \in \mathbb{Z}^m$. The statistical distance between $D_{\mathbb{Z}^m,\sigma}$ and $\boldsymbol{y} + D_{\mathbb{Z}^m,\sigma}$ is at most $\Delta\left(D_{\mathbb{Z}^m,\sigma}, \boldsymbol{y} + D_{\mathbb{Z}^m,\sigma}\right) \leq m\cdot\frac{\|\boldsymbol{y}\|_\infty}{\sigma}$.*

Lemma 2.10 ([AKPW13, Lemma 2.7]). *Let p, q be positive integers such that $p < q$. Given $R > 0$ an integer, the probability that there exists $e \in [-R, R]$ such that $\lfloor y\rfloor_p \neq \lfloor y + e\rfloor_p$, when $y \hookleftarrow U(\mathbb{Z}_q)$, is smaller than $\frac{2Rp}{q}$.*

Lemma 2.11. *If q is prime and $\mathcal{M}$ be a distribution over $\mathbb{Z}_q^{m\times n}$, and V a distribution over $\mathbb{Z}_q^n$ such that $\Delta\left(\mathcal{M}, U(\mathbb{Z}_q^{m\times n})\right) \leq \epsilon$. We have $\Delta\left(\mathcal{M}\cdot V, U(\mathbb{Z}_q^m)\right) \leq \epsilon + \alpha\cdot\left(1 - \frac{1}{q^m}\right)$, where $\alpha := \Pr[V = \mathbf{0}]$.*

Proof. Let $\mathbf{M} \leftarrow U(\mathbb{Z}_q^{m\times n})$. The statistical distance $\Delta\left(\mathbf{M}\cdot V, U(\mathbb{Z}_q^m)\right)$ equals

$$\frac{1}{2}\sum_{y\in\mathbb{Z}_q^m}\left|\Pr[\mathbf{M}V = y | V \neq \mathbf{0}]\cdot\Pr[V\neq\mathbf{0}] + \Pr[\mathbf{M}V = y | V = \mathbf{0}]\cdot\Pr[V = \mathbf{0}] - \frac{1}{q^m}\right|$$

By considering the sum for all terms such that $y \neq \mathbf{0}$ and then $y = \mathbf{0}$, we have

$$2\Delta\left(\mathbf{M}\cdot V, U(\mathbb{Z}_q^m)\right) = (q^m - 1)\left|\frac{1}{q^m}(1-\alpha) - \frac{1}{q^m}\right| + \left|\frac{1}{q^m}(1-\alpha) + \alpha - \frac{1}{q^m}\right|,$$

so that $\Delta\left(\mathbf{M}\cdot V, U(\mathbb{Z}_q^m)\right) \leq \alpha\left(1 - \frac{1}{q^m}\right)$. The claim follows from the triangle inequality:

$$\Delta\left(\mathcal{M}\cdot V, U(\mathbb{Z}_q^m)\right) \leq \Delta\left(\mathcal{M}\cdot V, \mathbf{M}\cdot V\right) + \Delta\left(\mathbf{M}\cdot V, U(\mathbb{Z}_q^m)\right).$$

□

Lemma 2.12. *Let M, m be positive integers, $M = m \cdot q + r$ with $0 \le r < m$. The statistical distance between the distributions $(U(\mathbb{Z}_M) \bmod m)$ and $U(\mathbb{Z}_m)$ is bounded by $\Delta(U(\mathbb{Z}_M) \bmod m, U(\mathbb{Z}_m)) \le \frac{r}{M}$.*

Proof. Let $M = mq+r$, with $0 \le r < m$. Observe that for $i \in \mathbb{Z}_m$ we can compute the number of integers of the form $i + jm$, smaller than $M-1$, by $\lfloor \frac{M-1-i}{m} \rfloor + 1$ which is also equal to $\lfloor q + \frac{r-1-i}{m} \rfloor + 1$. So the probability of getting $i \in \mathbb{Z}_m$ by sampling from $U(\mathbb{Z}_M) \mod m$ is equal to $\frac{q+1}{M}$ if $i < r$ or equal to $\frac{q}{M}$ if $i \ge r$. So the statistical distance that we want to evaluate is equal to:

$$\Delta = \frac{1}{2}\left(\sum_{i<r}\left|\frac{q+1}{M} - \frac{1}{m}\right| + \sum_{i\ge r}\left|\frac{q}{M} - \frac{1}{m}\right|\right) = \frac{r(m-r)}{Mm} \le \frac{r}{M}.$$

□

Lemma 2.13. *Let $a, b, c \in \mathbb{Z}$ such that $b > a$. We have $\Delta\left(U_{[a,b]}, U_{c+[a,b]}\right) \le \frac{|c|}{b-a}$, where $U_{[\alpha,\beta]}$ is the uniform distribution on $[\alpha, \beta] \cap \mathbb{Z}$.*

Lemma 2.14. *For any $\mathbf{A} \in \mathbb{R}^{m\times n}$, let $\alpha := \max_{i,j} |a_{i,j}|$. Then, we have the inequality $\det(\mathbf{A}\mathbf{A}^\top) \le (n \cdot \alpha^2)^m$.*

Proof. Since $\mathbf{A}\mathbf{A}^\top \in \mathbb{R}^{m\times m}$ is positive definite, we know that it has positive eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m \ge 0$. By the mean inequality, we have $\sqrt[m]{\lambda_1\lambda_2\cdots\lambda_m} \le \frac{\lambda_1+\cdots+\lambda_m}{m}$. This can be interpreted as $\det(\mathbf{A}\mathbf{A}^\top) \le \left(\frac{\mathrm{Tr}\mathbf{A}\mathbf{A}^\top}{m}\right)^m$ and the right hand side term can be bounded by $(n\alpha^2)^m$. □

Lemma 2.15. *Let $\ell_0 < \ell$ and a full rank matrix $\mathbf{Y} \in \mathbb{Z}^{\ell_0\times\ell}$ such that $|y_{ij}| \le Y$ and $\mathbf{z} \in \mathbb{Z}^{\ell_0}$. If the system $\mathbf{Y}\cdot\mathbf{x} = \mathbf{z}$ has an integer solution, then there exists an efficient algorithm that computes a solution $\mathbf{x} \in \mathbb{Z}^\ell$ such that:*

$$\|\mathbf{x}\|_\infty \le \|\mathbf{z}\|_\infty \cdot \ell_0 \cdot (Y\sqrt{\ell_0})^{\ell_0} + (\ell - \ell_0)\cdot(Y\sqrt{\ell_0})^{\ell_0}.$$

Proof. We assume w.l.o.g. that $\mathbf{Y} = [\mathbf{A}|\mathbf{B}]$, for a full rank matrix $\mathbf{A} \in \mathbb{Z}^{\ell_0\times\ell_0}$ and for some $\mathbf{B} \in \mathbb{Z}^{\ell_0\times(\ell-\ell_0)}$ such that $\max_{i,j}|a_{i,j}| \le Y$ and $\max_{i,j}|b_{i,j}| \le Y$. Denoting $\mathbf{x}^\top = [\mathbf{x}_0^\top, \mathbf{x}_1^\top]$, with $\mathbf{x}_0 \in \mathbb{Z}^{\ell_0}$ and $\mathbf{x}_1 \in \mathbb{Z}^{\ell-\ell_0}$, the system is equivalent to $\mathbf{A}\cdot\mathbf{x}_0 = \mathbf{z} - \mathbf{B}\cdot\mathbf{x}_1$ which is the same as

$$\mathbf{x}_0 = \frac{1}{\det\mathbf{A}} \cdot \left(\mathsf{adj}(\mathbf{A})\cdot\mathbf{z} - \mathsf{adj}(\mathbf{A})\cdot\mathbf{B}\cdot\mathbf{x}_1\right) \in \mathbb{Z}^{\ell_0},$$

where $\mathsf{adj}(\mathbf{A}) \in \mathbb{Z}^{\ell_0\times\ell_0}$ denotes the adjugate matrix. Since the system has an integer solution, there must exist $\mathbf{x}_1 \in \mathbb{Z}^{\ell-\ell_0}$ such that:

$$\mathsf{adj}(\mathbf{A})\cdot\mathbf{B}\cdot\mathbf{x}_1 = \mathsf{adj}(\mathbf{A})\cdot\mathbf{z} \bmod \det\mathbf{A}.$$

By solving the above modular linear system, we get a vector $\mathbf{x}_1 \in \mathbb{Z}^{\ell-\ell_0}$ such that $\|\mathbf{x}_1\|_\infty < |\det\mathbf{A}|$. By Lemma 2.14 we know that $|\det\mathbf{A}| \le (Y\sqrt{\ell_0})^{\ell_0}$, so $\|\mathbf{x}_1\|_\infty < (Y\sqrt{\ell_0})^{\ell_0}$.

Moreover, such an $\mathbf{x}_1 \in \mathbb{Z}^{\ell-\ell_0}$ implies that $\|\mathbf{x}_0\|_\infty \le \|\mathsf{adj}(\mathbf{A})\cdot\mathbf{z}\|_\infty + \|\mathsf{adj}(\mathbf{A})\cdot\mathbf{B}\|_\infty$. Since the entries of the matrix $\mathsf{adj}(\mathbf{A})$ are (ℓ_0-1)-minors of the matrix $\mathbf{A}$ we can use Lemma 2.14 to obtain: $\|\mathsf{adj}(\mathbf{A})\cdot\mathbf{z}\|_\infty \le \ell_0 \cdot \left(Y\cdot\sqrt{\ell_0-1}\right)^{\ell_0-1}\cdot\|\mathbf{z}\|_\infty$.

Notice that the entry at the intersection of row $i \in [\ell_0]$ and column $j \in [\ell-\ell_0]$ of the matrix $\mathsf{adj}(\mathbf{A})\cdot\mathbf{B} \in \mathbb{Z}^{\ell_0\times(\ell-\ell_0)}$ is equal to the determinant of the matrix $\mathbf{A}$ whose i-th column is replaced by the $\mathbf{b}_j \in \mathbb{Z}^{\ell_0}$, the j-th column in $\mathbf{B}$. Thus each entry of $\mathsf{adj}(\mathbf{A})\cdot\mathbf{B} \in \mathbb{Z}^{\ell_0\times(\ell-\ell_0)}$ is a $\ell_0\times\ell_0$ determinant. Thus, by Lemma 2.14, we also have the bound

$$\|\mathsf{adj}(\mathbf{A})\cdot\mathbf{B}\|_\infty \le (\ell-\ell_0)\cdot(\sqrt{\ell_0}Y)^{\ell_0}.$$

□

2.4 Admissible Hash Functions

Admissible hash functions were introduced by Boneh and Boyen [BB04] as a combinatorial tool for partitioning-based security proofs for which Freire *et al.* gave a simplified definition [FHPS13]. Jager [Jag15] considered the following generalization in order to simplify the analysis of reductions under decisional assumption.

Definition 2.1 ([Jag15])**.** *Let $\ell(\lambda), L(\lambda) \in \mathbb{N}$ be functions of a security parameter $\lambda \in \mathbb{N}$. Let $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$ be an efficiently computable function. For every $K \in \{0,1,\perp\}^L$, let the partitioning function $P_K : \{0,1\}^\ell \to \{0,1\}$ be defined as*

$$P_K(X) := \begin{cases} 0 & \text{if } \forall i \in [L] \ \ (\mathsf{AHF}(X)_i = K_i) \ \vee \ (K_i = \perp) \\ 1 & \text{otherwise} \end{cases}$$

*We say that AHF is a **balanced admissible hash function** if there exists an efficient algorithm $\mathsf{AdmSmp}(1^\lambda, Q, \delta)$ that takes as input $Q \in \mathsf{poly}(\lambda)$ and a non-negligible $\delta(\lambda) \in (0,1]$ and outputs a key $K \in \{0,1,\perp\}^L$ such that, for all $X^{(1)},\ldots,X^{(Q)}, X^\star \in \{0,1\}^\ell$ such that $X^\star \notin \{X^{(1)},\ldots,X^{(Q)}\}$, we have*

$$\gamma_{\max}(\lambda) \ge \Pr_K\left[P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \ \wedge \ P_K(X^\star) = 0\right] \ge \gamma_{\min}(\lambda),$$

where $\gamma_{\max}(\lambda)$ and $\gamma_{\min}(\lambda)$ are functions such that

$$\tau(\lambda) = \gamma_{\min}(\lambda)\cdot\delta(\lambda) - \frac{\gamma_{\max}(\lambda)-\gamma_{\min}(\lambda)}{2}$$

is a non-negligible function of λ.

Intuitively, the condition that $\tau(\lambda)$ be non-negligible requires $\gamma_{\min}(\lambda)$ to be noticeable and the difference of $\gamma_{\max}(\lambda)-\gamma_{\min}(\lambda)$ to be small.

It is known [Jag15] that balanced admissible hash functions exist for $\ell, L = \Theta(\lambda)$.

Theorem 2.16 ([Jag15, Theorem 1]). *Let $(C_\ell)_{\ell\in\mathbb{N}}$ be a family of codes $C_\ell : \{0,1\}^\ell \to \{0,1\}^L$ with minimal distance $c \cdot L$ for a constant $c \in (0,1/2)$. Then, $(C_\ell)_{\ell\in\mathbb{N}}$ is a family of balanced admissible hash functions. Moreover, $\mathsf{AdmSmp}(1^\lambda, Q, \delta)$ outputs a key $K \in \{0,1,\perp\}^L$ for which $\eta = \lfloor \frac{\ln(2Q+Q/\delta)}{-\ln((1-c))} \rfloor$ components are not $\perp$ and*

$$\gamma_{\max} \quad = \quad 2^{-\eta}, \qquad\qquad \gamma_{\min} = \big(1 - Q(1-c)\big)^\eta \cdot 2^{-\eta},$$

so that $\tau = (2\delta - (2\delta+1)\cdot Q\cdot(1-c)^\eta)/2^{\eta+1} \geq \delta$ is a non-negligible function of λ.

Lemma 2.17 ([KY16, Lemma 8],[ABB10, Lemma 28]). *Let $K \leftarrow \mathsf{AdmSmp}(1^\lambda, Q, \delta)$, an input space $\mathcal{X}$ and the mapping γ that maps a $(Q+1)$-uple $(X^\star, X_1, \ldots, X_Q)$ in $\mathcal{X}^{Q+1}$ to a probability value in $[0,1]$, given by:*

$$\gamma(X^\star, X_1, \ldots, X_Q) := \Pr_K\left[P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \ \wedge \ P_K(X^\star) = 0\right].$$

We consider the following experiment where we first execute the PRF security game, in which the adversary eventually outputs a guess $\hat{b} \in \{0,1\}$ of the challenger's bit $b \in \{0,1\}$ and wins with advantage ε. We denote by $X^\star \in \mathcal{X}$ the challenge input and $X_1, \ldots, X_Q \in \mathcal{X}$ the evaluation queries. At the end of the game, we flip a fair random coin $b'' \leftarrow\!\!\$\ U(\{0,1\})$. If the condition $P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \ \wedge \ P_K(X^\star) = 0$ is satisfied we define $b' = \hat{b}$. Otherwise, we define $b' = b''$. Then, we have

$$|\Pr[b' = b] - 1/2| \geq \gamma_{\min}\cdot\varepsilon - \frac{\gamma_{\max} - \gamma_{\min}}{2},$$

where $\gamma_{\min}$ and $\gamma_{\max}$ are the maximum and minimum of $\gamma(\mathbb{X})$ for any $\mathbb{X} \in \mathcal{X}^{Q+1}$.

2.5 (Deterministic) Randomness Extractors

Informally, a *randomness extractor* is a family of efficiently computable and efficiently sampleable hash functions $\mathcal{H}$, such that when we apply a uniformly sampled $h \leftarrow\!\!\$\ U(\mathcal{H})$ on a "random enough" source X, the resulting distribution, $h(X)$, is statistically close a the uniform distribution. An example of such a family is given by the Leftover Hash Lemma, which is given below.

Lemma 2.18. *Let integers m, n, ℓ such that $m > 2(n+\ell)\cdot\log q$, for some prime $q > 2$. Let $\mathbf{B}, \widetilde{\mathbf{B}} \leftarrow\!\!\$\ U(\mathbb{Z}_q^{m\times\ell})$ and $\mathbf{R} \leftarrow\!\!\$\ U(\{-1,1\}^{m\times m})$. For any matrix $\mathbf{F} \in \mathbb{Z}_q^{m\times n}$, the distributions $(\mathbf{B}, \mathbf{R}\cdot\mathbf{B}, \mathbf{R}\cdot\mathbf{F})$ and $(\mathbf{B}, \widetilde{\mathbf{B}}, \mathbf{R}\cdot\mathbf{F})$ are within $2^{-\Omega(n)}$ statistical distance.*

Notice that in the case of the Leftover Hash Lemma, we need to sample a fresh hash function for every new source that we want to use. Also the distribution of the source must be independent of the hash function. A *deterministic extractor* refers to a function that works as an extractor for a large collection of sources. Also the source may depend on the function, as long as it has the right amount of entropy. It turns out that a uniformly sampled function from a family of ξ-wise independent functions is a good deterministic extractor.

Definition 2.2. *A family* $\mathscr{F} \subset \left\{ f : \{0,1\}^{\bar{m}} \to \{0,1\}^{\bar{k}} \right\}$ *of efficiently sampleable and efficiently computable functions is called* ξ*-wise independent if for any* $x_1, x_2, \ldots, x_\xi \in \{0,1\}^{\bar{m}}$ *and* $y_1, y_2, \ldots y_\xi \in \{0,1\}^{\bar{k}}$ *we have:*

$$\Pr_{f \leftarrow U(\mathscr{F})} \left[(f(x_1) = y_1) \wedge (f(x_2) = y_2) \wedge \cdots \wedge (f(x_\xi) = y_\xi) \right] = \prod_{i=1}^{\xi} \Pr_{f \leftarrow U(\mathscr{F})} \left[f(x_i) = y_i \right]$$

It is well-known that ξ-wise independent functions can be obtained by choosing random polynomials of degree $\xi - 1$ over the finite field with $2^{\bar{m}}$ elements, $\mathbb{F}_{2^{\bar{m}}}$, (which cost $O(\xi \bar{m})$ bits to describe) and truncating their evaluations to their first $\bar{k}$ bits.

The following Lemma gives the concrete parameters when working with deterministic extractors for a finite collection $\mathscr{X}$ of distribution of high enough min-entropy.

Lemma 2.19 ([Dod00, Corollary 3])**.** *Fix any integers* $\bar{n}$*,* m*,* M*, any real* $\varepsilon < 1$ *and any collection* $\mathscr{X}$ *of* M *distributions over* $\{0,1\}^{\bar{m}}$ *of min-entropy* $\bar{n}$ *each. Define*

$$\xi = \bar{n} + \log M, \qquad \bar{k} = \bar{n} - \left(2 \log \frac{1}{\varepsilon} + \log\log M + \log \bar{n} + O(1) \right),$$

and let $\mathscr{F}$ *be any family of* ξ*-wise independent functions from* $\bar{m}$ *bits to* $\bar{k}$ *bits. With probability at least* $(1 - 1/M)$*, a random function* $f \leftarrow U(\mathscr{F})$ *is a good deterministic extractor for the collection* $\mathscr{X}$*. Namely, the distribution* $f(X)$ *is* ε*-close to* $U(\{0,1\}^{\bar{k}})$ *for any distribution* $X \in \mathscr{X}$*.*

$$\Pr_{f \leftarrow U(\mathscr{F})} \left[\Delta\left(f(X), U(\{0,1\}^{\bar{k}}) \right) \leq \epsilon \text{ for all } X \in \mathscr{X} \right] \geq 1 - \frac{1}{M}$$

2.6 Secret Sharing

In a secret sharing scheme, a dealer distributes individual shares to a set of shareholders such that certain qualified subsets can reconstruct the secret, while the forbidden sets learn nothing about it. We are concerned only with schemes where the collection of qualified sets, which is also called the access structure, is given by all the sets of cardinality greater than some given threshold t.

2.6.1 Shamir's Secret Sharing Scheme

For $t \leq N$ we recall the (t, N)-threshold secret sharing scheme from [Sha79], which is used to secret share $k \in \mathbb{F}_q$ among N players, such that any t of them can jointly reconstruct the secret, but any set less than t learn nothing about the secret.

To this end, the dealer samples a uniform polynomial $p(z) \in \mathbb{F}_q[z]$, of degree $t-1$ such that $p(0) = k$. The shares are computed as $k_i = p(i) \in \mathbb{F}_q$, for all $i \in [N]$.

We write $(k_1, k_2, \ldots, k_N) \leftarrow \mathsf{ShamirSS}(t, N, k)$ to specify that the k_i's are obtained as described above.

By polynomial interpolation, it is possible to reconstruct the degree $t-1$ secret polynomial $p(z) \in \mathbb{F}_q[z]$ from any subset of shares $\{k_i\}_{i\in\mathscr{S}}$, $|\mathscr{S}| = t$. In particular, we are able to recover the initial secret $k \in \mathbb{F}_q$, as follows:

- Compute the reconstruction coefficients $\lambda_{i,\mathscr{S}} \in \mathbb{F}_q$, that can be obtained just by knowing the set $\mathscr{S}$:
$$\lambda_{i,\mathscr{S}} = \prod_{k\in\mathscr{S}\setminus\{i\}} \frac{k}{k-i} \in \mathbb{F}_q$$
- Output $k := \sum_{i\in\mathscr{S}} k_i \cdot \lambda_{i,\mathscr{S}} \in \mathbb{F}_q$

2.6.2 Linear Integer Secret Sharing (LISS)

This section recalls the concept of linear integer secret sharing (LISS), as defined by Damgård and Thorbek [DT06]. The definitions below are taken from [Tho09] where the secret to be shared lives in an interval $[-2^l, 2^l]$ centered in 0, for some $l \in \mathbb{N}$.

Definition 2.3. *A **monotone** access structure on $[N]$ is a non-empty collection $\mathbb{A}$ of sets $A \subseteq [N]$ such that $\emptyset \notin \mathbb{A}$ and, for all $A \in \mathbb{A}$ and all sets B such that $A \subseteq B \subseteq [N]$, we have $B \in \mathbb{A}$. For an integer $t \in [N]$, the **threshold**-t access structure $T_{t,N}$ is the collection of sets $A \subseteq [N]$ such that $|A| \geq t$.*

Let $P = [N]$ be a set of shareholders. In a LISS scheme, a dealer D wants to share a secret s from a publicly known interval $[-2^l, 2^l]$. To this end, D uses a share generating matrix $\mathbf{M} \in \mathbb{Z}^{d\times e}$ and a random vector $\boldsymbol{\rho} = (s, \rho_2, \ldots, \rho_e)^\top$, where s is the secret to be shared $\{\rho_i\}_{i=2}^e$ are chosen uniformly in $[-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e$, for a large enough $l_0 \in \mathbb{N}$. The dealer D computes a vector $\boldsymbol{s} = (s_1, \ldots, s_d)^\top$ of share units as

$$\boldsymbol{s} = (s_1, \ldots, s_d)^\top = \mathbf{M} \cdot \boldsymbol{\rho} \in \mathbb{Z}^d.$$

Each party in $P = \{1, \ldots, N\}$ is assigned a set of share units. Letting $\psi : \{1, \ldots, d\} \to P$ be a surjective function, the i-th share unit s_i is assigned to the shareholder $\psi(i) \in P$, in which case player $\psi(i)$ is said to own the i-th row of $\mathbf{M}$. If $A \subseteq P$ is a set of shareholders, $M_A \in \mathbb{Z}^{d_A\times e}$ denotes the set of rows jointly owned by A. Likewise, $\boldsymbol{s}_A \in \mathbb{Z}^{d_A}$ denotes the restriction of $\boldsymbol{s} \in \mathbb{Z}^d$ to the coordinates jointly owned by the parties in A. The j-th shareholder's share consists of $\boldsymbol{s}_{\psi^{-1}(j)} \in \mathbb{Z}^{d_j}$, so that it receives $d_j = |\psi^{-1}(j)|$ out of the $d = \sum_{j=1}^n d_j$ share units. Sets $A \in \mathbb{A}$ are called *qualified* and $A \notin \mathbb{A}$ are called *forbidden*.

Definition 2.4 ([DT06])**.** *A* LISS *scheme is correct if the secret can be reconstructed from any set of shares $\{s_i : i \in A\}$ where A is a qualified set of shareholders. The secret is computed as a linear integer combination of the shares, with the coefficients that depend only on the set A.*

Definition 2.5 ([DT06])**.** *A* LISS *scheme is private if, for any two secrets s, s', any independent random coins $\boldsymbol{\rho} = (s, \rho_2, \ldots, \rho_e)$, $\boldsymbol{\rho}' = (s', \rho'_2, \ldots, \rho'_e)$ and any forbidden set A of shareholders, the distributions $s_A := M_A \cdot \boldsymbol{\rho}$ and $s'_A := M_A \cdot \boldsymbol{\rho}'$ are $2^{-\Omega(\lambda)}$ apart in terms of statistical distance.*

Damgård and Thorbek [DT06] showed how to construct LISS schemes from integer span programs [CF02].

Definition 2.6 ([CF02])**.** *An integer span program (ISP) is a tuple $\mathscr{M} = (\mathbf{M}, \psi, \boldsymbol{\varepsilon})$, where $\mathbf{M} \in \mathbb{Z}^{d \times e}$ is an integer matrix whose rows are labeled by a surjective function $\psi : \{1, \ldots, d\} \to \{1, \ldots, N\}$ and $\boldsymbol{\varepsilon} = (1, 0, \ldots, 0)^\top \in \mathbb{Z}^e$ is called target vector. The size of $\mathscr{M}$ is the number of rows d in $\mathbf{M}$.*

Definition 2.7. *Let Γ be a monotone access structure and let $\mathscr{M} = (\mathbf{M}, \psi, \boldsymbol{\varepsilon})$ an integer span program. Then, $\mathscr{M}$ is an ISP for Γ if it computes Γ: namely, for all $A \subseteq \{1, \ldots, N\}$, the following conditions hold:*

1. *If $A \in \Gamma$, there exists a reconstruction vector $\boldsymbol{\lambda}_A \in \mathbb{Z}^{d_A}$ such that $\boldsymbol{\lambda}_A^\top \cdot \mathbf{M}_A = \boldsymbol{\varepsilon}^\top$.*
2. *If $A \notin \Gamma$, there exists $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_e)^\top \in \mathbb{Z}^e$ such that $\mathbf{M}_A \cdot \boldsymbol{\kappa} = \mathbf{0} \in \mathbb{Z}^d$ and $\boldsymbol{\kappa}^\top \cdot \boldsymbol{\varepsilon} = 1$ (i.e., $\kappa_1 = 1$). In this case, the vector $\boldsymbol{\kappa}$ is called a sweeping vector for A.*

We also define $\kappa_{\max} = \max\{|a| \mid a \text{ is an entry in some sweeping vector}\}$.

Damgård and Thorbek showed [DT06] that, if we have an ISP $\mathscr{M} = (\mathbf{M}, \psi, \boldsymbol{\varepsilon})$ that computes the access structure Γ, a statistically private LISS scheme for Γ can be obtained by using $\mathbf{M}$ as the share generating matrix and setting $l_0 = l + \lceil \log_2(\kappa_{\max}(e-1)) \rceil + 1$, where l is the length of the secret.

A LISS scheme $\mathscr{L} = (\mathscr{M} = (\mathbf{M}, \psi, \boldsymbol{\varepsilon}), \Gamma, \mathscr{R}, \mathscr{K})$ is thus specified by an ISP for the access structure Γ, a space $\mathscr{R}$ of reconstruction vectors satisfying Condition 1 of Definition 2.7, and a space $\mathscr{K}$ of sweeping vectors satisfying Condition 2.

Lemma 2.20 ([Tho09, Lemma 3.1])**.** *Let $l_0 = l + \lceil \log_2(\kappa_{\max}(e-1)) \rceil + 1$. If $s \in [-2^l, 2^l]$ is the secret to be shared and $\boldsymbol{\rho}$ is randomly sampled from $[-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e$ conditionally on $\langle \boldsymbol{\rho}, \boldsymbol{\varepsilon} \rangle = s$, the LISS scheme derived from $\mathscr{M}$ is private. For any arbitrary $s, s' \in [-2^l, 2^l]$ and any forbidden set of shareholders $A \subset [N]$, the two distributions $\{s_A = M_A \cdot \boldsymbol{\rho} \mid \boldsymbol{\rho} \hookleftarrow U([-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e) \text{ s.t. } \langle \boldsymbol{\rho}, \boldsymbol{\varepsilon} \rangle = s\}$, and $\{s'_A = M_A \cdot \boldsymbol{\rho} \mid \boldsymbol{\rho} \hookleftarrow U([-2^{l_0+\lambda}, 2^{l_0+\lambda}]^e) \text{ s.t. } \langle \boldsymbol{\rho}, \boldsymbol{\varepsilon} \rangle = s'\}$ are within statistical distance $2^{-\lambda}$.*

Benaloh-Leichter LISS

Damgård and Thorbek [DT06] generalized the Benaloh-Leichter secret sharing scheme [BL88], which initially could work only in finite abelian groups, in order to use it over the integers. They gave an ISP construction based on Benaloh-Leichter for any access structure corresponding to monotone boolean formulas. As seen

above, this implies LISS schemes for such access structures. Since monotone access structures are in one-to-one correspondence with monotone boolean formulas, their construction implies LISS schemes for any monotone access structure, but not necessarily efficient schemes. The details of the construction can be found in [DT06]. We present here only its properties.

For an access structure given by some monotone formula f, there is an efficiently computable distribution matrix $\mathbf{M} \in \mathbb{Z}^{d\times e}$, with $d, e \leq \mathsf{size}(f)$. Moreover, the entries of the matrix $\mathbf{M}$ are only 0 and 1, i.e. $\mathbf{M} \in \{0,1\}^{d\times e}$ and the number of non-zero entries from each row is less than $\mathsf{depth}(f)$. The sweeping vectors live in $\{-1,0,1\}^e$.

Valiant's result [Val84] implies the existence of a monotone Boolean formula for the threshold-t function $T_{t,N}$, which has size $O(N^{5.3})$ and depth $O(\log N)$. This was improved by Hoory *et al.* [HHP06] who gave a monotone formula of size $O(N^{1+\sqrt{2}})$ and depth $O(\log N)$ for the majority function.[1]

In general, the size of one share unit is bounded by $l_0 + \lambda + \log\mathsf{depth}(f)$ bits. Since each player receives on average d/N share units, we estimate that the average size of a share to be $O\big(\mathsf{size}(f)/N \cdot (l_0 + \lambda + \log\mathsf{depth}(f))\big)$ bits. When we instantiate the scheme for the threshold function $T_{t,N}$ using the bounds from [HHP06], we get an average share size of $O\big(N^{\sqrt{2}} \cdot (l_0 + \lambda + \log\log N)\big)$ bits.

2.7 Hardness Assumptions

We define below the computational assumptions that we will use in the security proofs of this thesis.

2.7.1 Learning With Errors (LWE)

Definition 2.8 ([Reg05])**.** *Let $m \geq n \geq 1$, $q \geq 2$ and $\alpha \in (0,1)$ be functions of a security parameter λ. The* LWE *problem consists in distinguishing between the distributions $(\mathbf{A}, \mathbf{As}+\mathbf{e})$ and $U(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m)$, where $\mathbf{A} \sim U(\mathbb{Z}_q^{m\times n})$, $\mathbf{s} \sim U(\mathbb{Z}_q^n)$ and $\mathbf{e} \sim D_{\mathbb{Z}^m,\alpha q}$. For an algorithm $\mathscr{A} : \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m \to \{0,1\}$, we define:*

$$\mathbf{Adv}^{\mathsf{LWE}}_{q,m,n,\alpha}(\mathscr{A}) = |\Pr[\mathscr{A}(\mathbf{A}, \mathbf{As}+\mathbf{e}) = 1] - \Pr[\mathscr{A}(\mathbf{A}, \mathbf{u}) = 1]|,$$

where the probabilities are over $\mathbf{A} \sim U(\mathbb{Z}_q^{m\times n})$, $\mathbf{s} \sim U(\mathbb{Z}_q^n)$, $\mathbf{u} \sim U(\mathbb{Z}_q^m)$ and $\mathbf{e} \sim D_{\mathbb{Z}^m,\alpha q}$ and the internal randomness of $\mathscr{A}$. We say that $\mathsf{LWE}_{q,m,n,\alpha}$ is hard if for all PPT *algorithm $\mathscr{A}$, the advantage $\mathbf{Adv}^{\mathsf{LWE}}_{q,m,n,\alpha}(\mathscr{A})$ is negligible.*

[1] Note that a threshold-t function can be obtained from the majority function by fixing the desired number of input bits, so that we need a majority function of size $\leq 2N$ to construct a threshold function $T_{t,N}$.

2.7.2 Decisional Diffie Hellman (DDH)

Definition 2.9. *In a cyclic group $\mathbb{G}$ of prime order p, with a generator g, the Decision Diffie-Hellman Problem in $\mathbb{G}$, is to distinguish the following distributions:*

$$\{(g, g^a, g^b, g^{ab}) : a, b, c \leftarrow\!\!\!\rightarrow U(\mathbb{Z}_p)\} \text{ and } \{(g, g^a, g^b, g^c) : a, b, c \leftarrow\!\!\!\rightarrow U(Z_p)\}.$$

The Decision Diffie-Hellman assumption (DDH) in a group G, generated by g, is that the above distributions are computationally indistinguishable. For any PPT *adversary $\mathscr{A}$ we define the advantage:*

$$\mathbf{Adv}^{DDH}_{\mathscr{A}}(\lambda) = \left|\Pr[\mathscr{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathscr{A}(g, g^a, g^b, g^c) = 1]\right|,$$

where probabilities are taken over $a, b, c \leftarrow\!\!\!\rightarrow U(\mathbb{Z}_p)$ and the internal randomness of $\mathscr{A}$. The DDH assumption holds if for any PPT *adversary $\mathscr{A}$ the advantage is negligible.*

2.7.3 Decisional Composite Residuosity (DCR)

Definition 2.10 ([Pai99])**.** *Let p, q be prime numbers and $N = pq$. The Decision Composite Residuosity* (DCR) *assumption states that the following two distributions are computationally indistinguishable:*

$$\left\{t_0^N \bmod N^2 \mid t_0 \leftarrow U(\mathbb{Z}_N^\star)\right\} \overset{c}{\approx} \left\{t \mid t \leftarrow U(\mathbb{Z}_{N^2}^\star)\right\}$$

2.8 Pseudo-Random Functions (PRFs)

The standard security definition for a pseudo-random function requires that the adversary is unable to distinguish an oracle that always outputs function evaluations values from an oracle that outputs truly random values.

Definition 2.11 (Real-or-random security)**.** *Let λ be a security parameter. Let $F : \mathscr{K} \times \mathscr{X} \to \mathscr{Y}$ be an efficiently computable function and $\mathscr{Y}^{\mathscr{X}}$ be the set of all functions that map $\mathscr{X}$ to $\mathscr{Y}$. The advantage of a* PRF *distinguisher $\mathscr{A}$, making Q evaluation queries is defined as:*

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathscr{A},Q}(\lambda) := |\Pr[\mathscr{A}^{F(K,\cdot)}(1^\lambda) = 1 \mid K \leftarrow\!\!\!\rightarrow \mathscr{K}] - \Pr[\mathscr{A}^{R(\cdot)}(1^\lambda) = 1 \mid R \leftarrow\!\!\!\rightarrow U(\mathscr{Y}^{\mathscr{X}})]|,$$

where the probability is taken over all coin tosses. The function F is called pseudo-random if for any PPT *adversary $\mathscr{A}$ the advantage $\mathbf{Adv}^{\mathsf{PRF}}_{\mathscr{A},Q}(\lambda)$ is negligible.*

In some cases, it is convenient to work with the following equivalent definition.

Definition 2.12 (Find-then-guess security)**.** *Let λ be a security parameter. A function $F : \mathscr{K} \times \mathscr{X} \to \mathscr{Y}$ is a secure* PRF *if every* PPT *adversary $\mathscr{A}$ has only negligible advantage in the following game:*

Init. *The challenger initially chooses a random key $K \hookleftarrow \mathcal{K}$ and also flips a random coin $b \leftarrow U(\{0,1\})$.*

Queries 1. *On polynomially-many occasions, the adversary $\mathcal{A}$ chooses an arbitrary input $X \in \mathcal{X}$ and the challenger returns $F(K,X)$. These queries may be adaptive and depend on responses to previous queries.*

Challenge. *The adversary chooses an input $X^\star \in \mathcal{X}$ that differs from all queries of the previous stage. The challenger computes $Y_1 = F(K,X^\star)$ and randomly samples $Y_0 \hookleftarrow \mathcal{Y}$. Then, it returns Y_b to the adversary.*

Queries 2. *The adversary adaptively makes another series of queries for arbitrary inputs $X \neq X^\star$.*

Guess. *The adversary outputs a bit $\hat{b} \in \{0,1\}$ and wins if $\hat{b} = b$. Its advantage is defined as*

$$\mathbf{Adv}^{\mathrm{FG}}_{\mathcal{A}}(\lambda) := |\Pr[\hat{b} = b] - 1/2| = \frac{1}{2} \cdot |\Pr[\hat{b} = 1 \mid b = 1] - \Pr[\hat{b} = 1 \mid b = 0]|.$$

A standard hybrid argument over evaluation queries shows that – up to a multiplicative gap $\Theta(Q)$ between the advantage functions, where Q is the number of evaluation queries – find-then-guess security is equivalent to real-or-random security.

Chapter

3

Adaptively Secure Distributed PRFs from LWE

In this chapter we present our contribution from [LST18], which gives the first non-interactive construction of a Distributed Pseudo-Random Function family that simultaneously achieves security against adaptive corruptions. The security proof relies on the LWE assumption with super-polynomial modulus, without the use of random oracles. All lattice-based PRFs [BPR12, BLMR13, BP14], that use the rounding technique of [BPR12], suffer from this, including our construction.

The main insight of our result is a new security proof for the lattice-based Key-Homomorphic PRF of [BLMR13], which gives a reduction that knows the secret key of the PRF. This property carries over to the distributed case, where the reduction knows all the secret keys corresponding to each server. Hence, it can consistently answer both evaluation and corruption queries that the adversary adaptively makes. The result is not generic, as it cannot be applied to any Key-Homomorphic PRF. Instead we exploit the particular functionality of the [BLMR13] PRF.

Organization

In section 3.1 we give the formal syntax and security definitions for DPRFs. Next, in section 3.2 we show that the adaptive security model is strictly stronger than the static corruption model, emphasizing that obtaining a DPRF construction secure under adaptive corruptions is non-trivial. Section 3.3 presents the starting point of our results, namely the generic construction based on KH-PRFs from [BLMR13], that gives statically secure DPRFs. Before we present the adaptively secure DPRF in section 3.5.1, we first discuss its centralized version, for a better understanding of our main results. We end the discussion about DPRFs with section 3.6 by showing how to use homomorphic signatures to upgrade the scheme to satisfy robustness against faulty servers.

3.1 Definitions

In this section we give the DPRF syntax along with both the adaptive and the static security model.

3.1.1 Distributed Pseudo-Random Functions (DPRFs)

Definition 3.1 (DPRFs)**.** *A distributed pseudo-random function (DPRF) is a tuple of efficient algorithms* (Setup, Keygen, Share, PEval, Eval, Combine) *with the following specification.*

Setup$(1^\lambda, 1^\ell, 1^t, 1^N)$**:** *Takes as input a security parameter* 1^λ*, a number of servers* 1^N*, a threshold* 1^t *and a desired input length* 1^ℓ *and outputs public parameters* pp*. It also specifies the key space* $\mathcal{K}$*, the domain* $\mathcal{X} := \{0,1\}^\ell$ *and the range* $\mathcal{Y}$ *of the evaluation function.*

Keygen(pp)**:** *Takes as input the public parameters* pp *and returns a secret evaluation key* $SK \in \mathcal{K}$*.*

Share(pp, SK)**:** *The key sharing algorithm inputs a random master secret key* $SK \in \mathcal{K}$ *and outputs a tuple of shares* $(SK_1, \ldots, SK_N) \in \mathcal{K}^N$*, which form a* (t,N)*-threshold secret sharing of* SK*.*

PEval(pp, SK_i, X)**:** *Takes as input the public parameters, a share of the secret key and input* $X \in \mathcal{X}$*. It outputs a partial evaluation* $Y_i \in \mathcal{Y}$*.*

Eval(pp, SK, X)**:** *Takes as input the public parameters, the master secret key and input* $X \in \mathcal{X}$*. It outputs the evaluation* $Y \in \mathcal{Y}$*.*

Combine$(\mathcal{S}, (Y_{i_1}, Y_{i_2}, \ldots, Y_{i_t}))$**:** *Takes as input a* t*-subset* $\mathcal{S} \subset [N]$*, a set* $\{Y_i\}_{i \in \mathcal{S}}$ *of* t *partial evaluations and returns an evaluation* $Y \in \mathcal{Y}$*.*

Correctness. We require that for $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$, $SK \leftarrow \mathsf{Keygen}(\mathsf{pp})$, $(SK_1, SK_2, \ldots, SK_N) \leftarrow \mathsf{Share}(\mathsf{pp}, SK)$, for any message $X \in \mathscr{X}$, $Y \leftarrow \mathsf{Eval}(\mathsf{pp}, SK, X)$, $\{Y_j \leftarrow \mathsf{PEval}(\mathsf{pp}, SK_j, X)\}_{j=1}^N$ and any subset $\mathscr{S} \subset [N]$ such that $|\mathscr{S}| = t$ there exists a negligible function negl such that:

$$\Pr[\mathsf{Combine}(\mathscr{S}, \{Y_i\}_{i\in\mathscr{S}}) \neq Y] \leq \mathsf{negl}(\lambda)$$

3.1.2 Security

We say that a DPRF provides *adaptive security* if it remains secure against an adversary that can corrupt users in an adaptive manner. In particular, the adversary can arbitrarily interleave evaluation and corruption queries. The formal definition is the following.

Definition 3.2 (Adaptive DPRF security)**.** *Let λ be a security parameter and let integers $\ell, t, N \in \mathsf{poly}(\lambda)$. We say that the DPRF is secure under adaptive corruptions if no* PPT *adversary has non-negligible advantage in the following game:*

1. *The challenger generates $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$ and a secret key $SK \leftarrow \mathsf{Keygen}(\mathsf{pp})$, which is broken into N shares $(SK_1, \ldots, SK_N) \leftarrow \mathsf{Share}(SK)$. It also initializes the empty set $\mathscr{C} \leftarrow \emptyset$ and flips a random coin $b \hookleftarrow U(\{0,1\})$. The public parameters are given to the adversary.*

2. *The adversary $\mathscr{A}$ adaptively interleaves the following kinds of queries.*

   Corruption: *$\mathscr{A}$ chooses an index $i \in [N]\setminus\mathscr{C}$. The challenger returns SK_i to $\mathscr{A}$ and sets $\mathscr{C} := \mathscr{C} \cup \{i\}$.*

   It is required that $|\mathscr{C}| < t$ at any time.

   Evaluation: *$\mathscr{A}$ chooses a pair $(i, X) \in [N] \times \mathscr{X}$, the challenger returns $Y_i = \mathsf{PEval}(SK_i, X)$.*

3. *The adversary chooses an input $X^\star$ such that $|\mathscr{C}^\star| < t$, where $\mathscr{C}^\star$ is the set of indices that were corrupted by $\mathscr{A}$ or such that an evaluation query of the form $(i, X^\star)$ was made. At this point, the challenger randomly samples $Y_0^\star \hookleftarrow U(\mathscr{Y})$ and computes $Y_1^\star = \mathsf{Eval}(SK, X^\star)$. Then, it returns $Y_b^\star$ to the adversary.*

4. *The adversary $\mathscr{A}$ adaptively makes more queries as in Stage 2 under the restriction that, at any time, we should have $|\mathscr{C}^\star| < t$.*

5. *$\mathscr{A}$ outputs a bit $\hat{b} \in \{0,1\}$ and wins if $\hat{b} = b$. Its advantage is defined to be $\mathbf{Adv}_{\mathscr{A}}^{\mathrm{DPRF}}(\lambda) := |\Pr[\hat{b} = b] - 1/2|$.*

We also recall the *static corruption* security model, that was used in [BLMR13], in which the adversary has to announce the set of corruptions at the start of the attack, before making any evaluation queries.

Definition 3.3 (Static DPRF security)**.** *Let λ be a security parameter and let integers $\ell, t, N \in \mathsf{poly}(\lambda)$. We say that the DPRF is pseudo-random under static corruptions if no* PPT *adversary has non-negligible advantage in the following game:*

1. *The challenger generates* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$ *and a secret key* $SK \leftarrow \mathsf{Keygen}(\mathsf{pp})$*, which is broken into N shares* $(SK_1, \ldots, SK_N) \leftarrow \mathsf{Share}(SK)$*. The public parameters are given to the adversary. It also initializes the empty set* $\mathcal{V} \leftarrow \emptyset$ *and flips a random coin* $b \hookleftarrow U(\{0,1\})$*.*

2. *The adversary* $\mathscr{A}$ *chooses the set of corruptions* $\mathscr{C}^\star := \{i_1, i_2, \ldots, i_{t-1}\}$ *and the challenger responds with* $\{SK_{i_1}, SK_{i_2}, \ldots, SK_{i_{t-1}}\}$*.*

3. *The adversary adaptively makes evaluation queries X to which the challenger answers with* $\{Y_i = \mathsf{PEval}(SK_i, X)\}_{i \in [N] \setminus \mathscr{C}^\star}$ *and sets* $\mathcal{V} := \mathcal{V} \cup \{X\}$*.*

4. *The adversary chooses an input* $X^\star$ *that was never queried before, i.e.* $X^\star \notin \mathcal{V}$*. At this point, the challenger randomly samples* $Y_0^\star \hookleftarrow U(\mathscr{Y})$ *and computes* $Y_1^\star = \mathsf{Eval}(SK, X^\star)$*. Then, it returns* $Y_b^\star$ *to the adversary.*

5. *The adversary* $\mathscr{A}$ *adaptively makes more queries as in Stage* 3 *under the restriction that, at any time, we should have* $X^\star \notin \mathcal{V}$*.*

6. $\mathscr{A}$ *outputs a bit* $\hat{b} \in \{0,1\}$ *and wins if* $\hat{b} = b$*. Its advantage is defined to be* $\mathbf{Adv}_{\mathscr{A}}^{\mathrm{DPRF}}(\lambda) := |\Pr[\hat{b} = b] - 1/2|$*.*

3.2 Adaptive vs Static Corruptions

This section gives a separation between Definition 3.2 and the definition of static security used in [BLMR13] (which is recalled in Definition 3.3). It is well-known that static security does not imply adaptive security in distributed threshold protocols (see, e.g., [CDD+99]). In the case of DPRFs, we show that allowing even a *single* evaluation query before any corruption query already gives a stronger game-based definition than Definition 3.3.

Theorem 3.1. *For any $t, N \in \mathsf{poly}(\lambda)$ such that $t < N/2$, there exists a DPRF which is secure in the sense of Definition 3.3 but insecure in the sense of Definition 3.2.*

Proof. Let $\Pi = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Share}, \mathsf{PEval}, \mathsf{Eval}, \mathsf{Combine})$ be a DPRF family which provides static security as captured by Definition 3.3. We assume that secret keys live in a large finite field $\mathbb{F}_p$ (the key-homomorphic-based constructions implied by [BLMR13, BP14] can be modified to satisfy this). We modify the DPRF family Π into:

$\Pi^* = (\mathsf{Setup}^*, \mathsf{Keygen}^*, \mathsf{Share}^*, \mathsf{PEval}^*, \mathsf{Eval}^*, \mathsf{Combine}^*)$ which is insecure in the experiment of Definition 3.2 but remains secure under Definition 3.3 for the same threshold t. The construction Π^* goes as follows.

Setup$^*(1^\lambda, 1^\ell, 1^t, 1^N)$**:** Run $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$ and output pp.

Keygen$^*(\mathsf{pp})$**:** Run $SK \leftarrow \mathsf{Keygen}(\mathsf{pp})$.

Share$^*(\mathsf{pp}, SK)$**:** Given $SK = k$, conduct the following steps.

1. Run $(k_1, \ldots, k_N) \leftarrow \mathsf{Share}(\mathsf{pp}, k)$.
2. Generate an independent $(t-1, N)$ Shamir secret sharing of k and let $(k'_1, \ldots, k'_N)$ be the resulting shares.
3. Choose a random subset $\mathcal{T} \subset [N]$ of cardinality $|\mathcal{T}| = N - t + 1$. Choose uniformly random vector $(k''_1, \ldots, k''_N) \leftarrow\!\!\!\rightarrow U(\mathbb{F}_p^N)$ and define $(\tilde{k}_1, \ldots, \tilde{k}_N)$ in the following way

$$\tilde{k}_i := \begin{cases} k'_i & \text{if } i \notin \mathcal{T} \\ k''_i & \text{if } i \in \mathcal{T} \end{cases}$$

   For each $i \in [N]$ define the i-th secret key share as $SK_i := \big(k_i, \tilde{k}_i, \mathcal{T}\big)$.

Output $(SK_1, \ldots, SK_N)$.

PEval$^*(\mathsf{pp}, SK_i, X)$**:** Given the secret key share $SK_i = \big(k_i, \tilde{k}_i, \mathcal{T}\big)$, compute $F_{k_i}(X) = \mathsf{PEval}(\mathsf{pp}, k_i, X)$ and output $Y_i = \big(F_{k_i}(X), \mathcal{T}\big)$.

Eval$^*(\mathsf{pp}, SK, X)$**:** Compute $Y \leftarrow \mathsf{Eval}(\mathsf{pp}, k, X)$ and output Y.

Combine$^*(\mathcal{S}, (Y_{i_1}, \ldots, Y_{i_t}))$**:** For each $\kappa \in [t]$, parse Y_{i_κ} as $Y_{i_\kappa} = \big(\tilde{Y}_{i_\kappa}, \mathcal{T}\big)$, where $\tilde{Y}_{i_\kappa} = F_{k_{i_\kappa}}(X)$. Run $Y \leftarrow \mathsf{Combine}(\mathcal{S}, (\tilde{Y}_{i_1}, \ldots, \tilde{Y}_{i_t}))$ and return Y.

It is easy to see that Π^* is trivially insecure in the sense of Definition 3.2. Before corrupting any server, the adversary can make a single evaluation query on an arbitrary server to learn $\mathcal{T}$. By corrupting all servers $i \in [N] \setminus \mathcal{T}$, the adversary can then obtain $t-1$ shares of $SK = k$ so as to reconstruct it.

The proof that Π^* remains statically secure relies on the idea that, in the game of Definition 3.3, the adversary has to choose the set $\mathcal{S}^\star$ of corrupted servers *before* making any evaluation query. Hence, with overwhelming probability $1 - 1/\binom{N}{t-1} > 1 - \left(\frac{t}{N-t}\right)^{t-1}$, we have $\mathcal{S}^\star \neq [N] \setminus \mathcal{T}$, in which case $\{\tilde{k}_i\}_{i \in \mathcal{S}^\star}$ is information-theoretically independent of SK. We give a reduction below.

Assuming that we have a DPRF adversary $\mathcal{A}$ against Π^*, we build a DPRF adversary $\mathcal{B}$ against Π as follows. First $\mathcal{B}$ relays to $\mathcal{A}$ the public parameters that are received from its challenger $\mathcal{C}$. Then, $\mathcal{B}$ chooses a random subset $\mathcal{T} \subset [N]$ of cardinality $|\mathcal{T}| = N - t + 1$. Then, $\mathcal{B}$ starts interacting with $\mathcal{A}$ that chooses an arbitrary $(t-1)$-subset $\mathcal{S}^\star = \{i_1^\star, \ldots, i_{t-1}^\star\} \in [N]$. If $\mathcal{S}^\star = [N] \setminus \mathcal{T}$, $\mathcal{B}$ halts and the attack is unsuccessful. Otherwise, we have $|\mathcal{S}^\star \cap \mathcal{T}| \geq 1$, which means that at least one of the $\{\tilde{k}_{i_\kappa^\star}\}_{\kappa=1}^{t-1}$ is supposed to have been obtained by replacing a coordinate of $(k'_{i_1^\star}, \ldots, k'_{i_{t-1}^\star})$ with a random element of $\mathbb{F}_p$. Since $\{k'_{i_\kappa^\star}\}_{\kappa=1}^{t-1}$ form a Shamir secret

sharing over a finite field, replacing at least one of the shares $\{k'_{i^\star_\kappa}\}_{\kappa=1}^{t-1}$ with a random value results in a randomly distributed $(\tilde{k}_{i^\star_1},\ldots,\tilde{k}_{i^\star_{t-1}})$ which is uncorrelated to k.

In this case, $\mathscr{B}$ sends $\mathscr{S}^\star$ to its own DPRF challenger and obtains the key share $k_{i^\star_1},\ldots,k_{i^\star_{t-1}}$. It also chooses $(\tilde{k}_{i^\star_1},\ldots,\tilde{k}_{i^\star_{t-1}}) \hookleftarrow U(\mathbb{F}_p^{t-1})$ uniformly and gives $\{(k_{i^\star_\kappa},\tilde{k}_{i^\star_\kappa},\mathscr{T})\}_{\kappa=1}^{t-1}$ to $\mathscr{A}$. Then, $\mathscr{B}$ answers $\mathscr{A}$'s evaluation queries by invoking its own challenger on the same inputs and constructing the responses in the obvious way. At the end of the experiment, $\mathscr{B}$ outputs whatever $\mathscr{A}$ outputs.

If ϵ denotes the advantage of $\mathscr{A}$ and Guess denotes the event that $\mathscr{S}^\star = [N]\backslash\mathscr{T}$, we know that $\Pr[\mathscr{B}\text{ wins}|\neg\mathsf{Guess}] = \frac{1}{2}+\epsilon$. This implies

$$\begin{aligned}
\Pr[\mathscr{B}\text{ wins}] &= \Pr[\mathscr{B}\text{ wins} \mid \mathsf{Guess}]\cdot\Pr[\mathsf{Guess}] + \Pr[\mathscr{B}\text{ wins} \mid \neg\mathsf{Guess}]\cdot\Pr[\neg\mathsf{Guess}] \\
&> \Pr[\mathscr{B}\text{ wins} \mid \neg\mathsf{Guess}]\cdot\Pr[\neg\mathsf{Guess}] \\
&= \left(\frac{1}{2}+\epsilon\right)\cdot(1-\Pr[\mathsf{Guess}]) \\
&> \left(\frac{1}{2}+\epsilon\right)\cdot\left(1-\left(\frac{t}{N-t}\right)^{t-1}\right) > \left(\frac{1}{2}+\epsilon\right)-\left(\frac{t}{N-t}\right)^{t-1}.
\end{aligned}$$

Since $t, N \in \mathsf{poly}(\lambda)$, we have $\left(\frac{t}{N-t}\right)^{t-1} < 2^{-\Omega(\lambda)}$. Consequently, if $\epsilon > 0$ is noticeable, so is $|\Pr[\mathscr{B}\text{ wins}] - \frac{1}{2}| > \epsilon - 2^{-\Omega(\lambda)}$. □ □

Note that the above separation still holds for small non-constant values of t and N if we assume polynomial or slightly super-polynomial adversaries. Indeed, if $t = \Theta(\log\lambda)$ and $N = \Theta(\log^2\lambda)$, for example, the adversary's probability to guess $\mathscr{T}$ is $1/\binom{N}{t-1} < \left(\frac{t}{N-t}\right)^{t-1}$, which is roughly $1/(\log\lambda)^{\log\lambda} = \lambda^{-\omega(1)}$. For any constant $c > 0$, if $t = \Theta(\lambda^{1/c})$ and $N = \Theta(\lambda^{2/c})$, the guessing probability becomes sub-exponentially small.

3.3 DPRF from KH-PRF

In this section we recall the generic DPRF construction that is obtained from Key-Homomorphic PRFs, given in [BLMR13]. We also remark that this modular approach can only achieve static security.

3.3.1 Key-homomorphic PRFs

Informally, a key-homomorphic PRF is a secure pseudo-random function family $F : \mathscr{K}\times\mathscr{X}\to\mathscr{Y}$ that is homomorphic with respect to the key space i.e., given $F(K_1,X)$ and $F(K_2,X)$ there is an efficient algorithm that computes $F(K_1\oplus K_2,X)$. Here we assumed $\mathscr{K}$ has a group structure with respect to the operation $\oplus$. For simplicity assume also that the range $\mathscr{Y}$ is endowed with group structure $\otimes$ as well. Next we give the formal definition from [BLMR13].

Definition 3.4. *Let $F : \mathscr{K}\times\mathscr{X}\to\mathscr{Y}$ be an efficiently computable function such that:*

- *F is a secure* PRF
- *for any $k_1, k_2 \in \mathcal{K}$ and any $X \in \mathcal{X}$ we have*

$$F(k_1 \oplus k_2, X) = F(k_1, X) \otimes F(k_2, X)$$

When the homomorphic property is not exact, i.e. $F(k_1 \oplus k_2, X)$ needs only to be sufficiently "close" to $F(k_1, X) \otimes F(k_2, X)$ we say that the PRF is *almost* key-homomorphic. Formally, we give the definition below for the particular key space $\mathcal{K} = \mathbb{Z}_q^m$ and range $\mathcal{Y} = \mathbb{Z}_p^m$, where $p < q$ are integers.

Definition 3.5. *An efficiently computable function $F : \mathbb{Z}_q^m \times \mathcal{X} \to \mathbb{Z}_p^m$ is called an γ-almost key-homomorphic* PRF *if:*

- *F is a secure* PRF
- *for any $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{Z}_q^m$ and any $X \in \mathcal{X}$ we have:*

$$F(\mathbf{k}_1 + \mathbf{k}_2, X) = F(\mathbf{k}_1, X) + F(\mathbf{k}_2, X) + \mathbf{e} \text{ such that the error } \|\mathbf{e}\|_\infty \leq \gamma.$$

Naor, Pinaks and Reingold have given the following simple DPRF construction in [NPR00]. Assume $H : \mathcal{X} \to \mathbb{G}$ is a hash function that maps the set $\mathcal{X}$ to a finite cyclic group $(\mathbb{G}, \cdot)$ of prime order q. Define the function $F_{DDH} : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{G}$ as:

$$F_{DDH}(k, X) := H(X)^k \in \mathbb{G}$$

The key-homomorphic property $F_{DDH}(k_1 + k_2, X) = F_{DDH}(k_1, X) \cdot F_{DDH}(k_2, X)$ is true as well. The function F_{DDH} was proved to be a secure DPRF in the random oracle model, assuming the DDH assumption holds in the multiplicative group $\mathbb{G}$ [NPR00].

The BLMR Key-Homomorphic PRF

The first (almost) key-homomorphic PRF from standard assumptions was proposed by Boneh, Lewi, Montgomery and Raghunathan in [BLMR13]. The security of the construction was proved under the LWE assumption with super-polynomial approximation factors. We will briefly recall the construction below.

Let m, n, p, q be integers such that $m = n \cdot \lceil \log q \rceil$ and p divides q. Sample full rank binary matrices $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \{0,1\}^{m \times m}$. For any key $\mathbf{k} \in \mathbb{Z}_q^m$ and any binary input $X = x_1 x_2 \ldots x_\ell \in \{0,1\}^\ell$, define $F_{LWE} : \mathbb{Z}_q^m \times \{0,1\}^\ell \to \mathbb{Z}_p^m$ as:

$$F_{LWE}(\mathbf{k}, X) = \left\lfloor \prod_{i=1}^{\ell} \mathbf{A}_{x_i} \cdot \mathbf{k} \right\rfloor_p$$

where $\lfloor z \rceil_p := \lfloor z \cdot p/q \rceil \in \mathbb{Z}_p$ for any $z \in \mathbb{Z}_q$, is the rounding function introduced in [BPR12]. Because of the rounding we don't have an exact key-homomorphic property, but an *almost key-homomorphic* property [BLMR13]: for any keys $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{Z}_q^m$ and any input $X \in \{0,1\}^\ell$ we have:

$$F_{LWE}(\mathbf{k}_1 + \mathbf{k}_2, X) = F_{LWE}(\mathbf{k}_1, X) + F_{LWE}(\mathbf{k}_2, X) + \mathsf{err}$$

for some small error $\mathsf{err} \in \{0,1\}^m$.

A more efficient variant of the above almost key-homomorphic PRF was later given in [BP14], under an even weaker LWE assumption.

3.3.2 The Generic Construction

Assume for some input length ℓ there is a secure key-homomorphic PRF, $F : \mathbb{F}_q \times \mathscr{X} \to \mathscr{Y}$, with input space $\mathscr{X} = \{0,1\}^\ell$, key space $\mathbb{F}_q$, and the set $\mathscr{Y}$ has a group structure with respect to some operation $\otimes : \mathscr{Y} \times \mathscr{Y} \to \mathscr{Y}$. Recall from Definition 3.4 that for any keys $k_1, k_2 \in \mathbb{F}_q$ and any input $X \in \mathscr{X}$ we must have

$$F(k_1 + k_2, X) = F(k_1, X) \otimes F(k_2, X)$$

In order to obtain a t-threshold DPRF for N users, we use Shamir's Secret Sharing scheme recalled in Section 2.6.1. The idea is to (t, N)-secret share the evaluation key $k \in \mathbb{F}_q$ to get $(k_1, k_2, \ldots, k_N)$ and compute partial evaluations as $F(k_i, X)$. Notice that we can recover the evaluation $F(k, X)$ of the PRF from any set of partial evaluations $\{F(k_i, X)\}_{i \in \mathscr{S}}$, $\mathscr{S} \subset [N]$, $|\mathscr{S}| = t$, using the public reconstruction coefficients as:

$$F(k, X) = \sum_{i \in \mathscr{S}} \lambda_{i,\mathscr{S}} \cdot F(k_i, X)$$

Given a secure key-homomorphic PRF family $F : \mathbb{F}_q \times \mathscr{X} \to \mathscr{Y}$, we give the formal construction by the following algorithms:

Setup$(1^\lambda, 1^\ell, 1^t, 1^N)$**:** Outputs the public parameters, pp, used by $F : \mathbb{F}_q \times \mathscr{X} \to \mathscr{Y}$, the key-homomorphic PRF.

Keygen(pp)**:** Samples a uniform key $k \leftarrow\!\!\$\, U(\mathbb{F}_q)$ and sets $SK := k$.

Share(pp, SK)**:** Given the $SK = k$, it runs $(k_1, k_2, \ldots, k_N) \leftarrow \mathsf{ShamirSS}(t, N, k)$ and sets $SK_i := k_i$ for all $i \in [N]$.

PEval(pp, SK_j, X)**:** For $SK_i = k_i$ and $X \in \mathscr{X}$, the algorithm outputs $Y_i := F(k_i, X)$.

Eval(pp, SK, X)**:** For $SK = k$, $X \in \mathscr{X}$, outputs $F(k, X)$.

Combine($\mathcal{S}$, $\{Y_i\}_{i\in\mathcal{S}}$): Given the set $\mathcal{S} \subset [N]$ such that $|\mathcal{S}| = t$ and the partial evaluations $\{Y_i\}_{i\in\mathcal{S}}$, the algorithm computes the reconstruction coefficients $\lambda_{i,\mathcal{S}}$ (as in Section 2.6.1), which are publicly computable. Outputs

$$Y := \sum_{i\in\mathcal{S}} \lambda_{i,\mathcal{S}} \cdot Y_i$$

Theorem 3.2. *[BLMR13, Theorem 7.2] If F is a secure Key-Homomorphic* PRF*, then the above DPRF construction is statically secure (Definition 3.3).*

The PRF that was proposed in [BLMR13] is actually *almost key-homomorphic*, meaning that the homomorhic property holds up to a small error term. The same work shows how to adapt the construction and the proof to work in the case of an *γ-almost key-homomorphic* PRF. Recall that a γ-almost key-homomorphic PRF (Def. 3.5) refers to an efficiently computable $F : \mathbb{Z}_q \times \mathcal{X} \to \mathbb{Z}_p$ such that for any $k_1, k_2 \in \mathbb{Z}_q$ any $X \in \mathcal{X}$ there is a small error term $e \in \mathbb{Z}_p$ such that $|e| \leq \gamma$ and:

$$F(k_1 + k_2, X) = F(k_1, X) + F(k_2, X) + e$$

The issue that has to be fixed for this variant is that when we do the reconstruction from the partial evaluations we get an error term because the homomorphic property of the PRF is not exact.

To make this work we would need to round off the error at the end of the computation. One problem that occurs when using the Lagrange coefficients $\lambda_{i,\mathcal{S}}$ as elements from $\mathbb{Z}_p$, is that they do not remain small relative to p, so we cannot apply the rounding successfully. To solve this problem, the authors of [BLMR13] used the technique of "clearing the denominators" [ABV+12]. The observation is that $N! \cdot \lambda_{i,\mathcal{S}} \in \mathbb{Z}$ for all reconstruction coefficients, thus their size does not depend on p anymore. So the combining algorithm multiplies all the reconstruction coefficients by $N!$. By doing this, some control over the size of the error is gained. Now we can compute a bound, independent of p, on the final error term. Therefore for the reconstruction to work correctly we would need a rounding parameter u such that the error incurred by applying the γ-almost key-homomorphic PRF can be correctly rounded off. Notice that

$$N! \cdot F(k, X) = \sum_{i\in\mathcal{S}} N! \cdot \lambda_{i,\mathcal{S}} F(k_i, X) + \mathsf{err}$$

where err can be bounded by $|\mathsf{err}| \leq \gamma \cdot |\mathcal{S}| \cdot \max_{i\in\mathcal{S}} |N! \cdot \lambda_{i,\mathcal{S}}| \leq \gamma t (N!)^2$

By Lemma 2.10, the final error can be removed for as long as $p/u > 2\gamma t \cdot (N!)^2$, thus making the reconstruction possible.

3.4 A Centralized Version

Before we present the DPRF construction in section 3.5, we believe it is instructive to first discuss its centralized version, which can be seen as a variant of the

key-homomorphic PRF proposed by Boneh *et al.* [BLMR13] and later improved by Banerjee and Peikert [BP14].

3.4.1 Overview

Despite the resemblance with the above-mentioned PRFs, the security proof is very different in that it does not use a hybrid argument over the input bits. Instead, we partition the input space (analogously to proof techniques for, e.g., identity-based encryption [Wat05]) and use the hardness assumption through the lossy mode of LWE [GKPV10].

One important characteristic of our centralized construction is that, unlike the ones from [BLMR13] or [BP14], in our new security proof the reduction always has access to the secret evaluation keys of the PRF. This is a very useful feature when analyzing the decentralized version, because now the reduction is able to easily answer adaptive evaluations and corruption queries in a consistent manner. These ideas have been used to construct standard-model adaptively secure Constrained PRFs as well [DKN+20].

Recall from subsection 3.3.1 that in [BLMR13, BP14], a PRF evaluation of an input x is of the form $\mathbf{y} = \lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rfloor_p \in \mathbb{Z}_p^m$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret key and $\mathbf{A}(x) \in \mathbb{Z}_q^{n \times m}$ is an input-dependent matrix obtained from public matrices $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$. Our variant is similar at a high level, with two main differences. First, we derive $\mathbf{A}(x)$ from a set of $2L$ public matrices $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^L$. Second, $\lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rfloor_p$ is not quite our PRF evaluation. Instead, we obtain the PRF value by using $\mathbf{z} := \lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rfloor_p$ as a source of entropy for a deterministic randomness extractor π. So the actual PRF output is $\mathbf{y} := \pi(\mathbf{z})$.

The security proof departs from [BLMR13, BP14] by exploiting the connection between the schemes and the Gentry-Sahai-Waters FHE [GSW13]. For each $i \in [L]$ and $b \in \{0,1\}$, we interpret the matrix $\mathbf{A}_{i,b} \in \mathbb{Z}_q^{n \times m}$ as a GSW ciphertext $\mathbf{A}_{i,b} = \mathbf{A} \cdot \mathbf{R}_{i,b} + \mu_{i,b} \cdot \mathbf{G}$, where $\mathbf{R}_{i,b} \in \{-1,1\}^{m \times m}$, $\mu_{i,b} \in \{0,1\}$ and $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is the gadget matrix of [MP12]. Before evaluating the PRF on an input X, we encode $X \in \{0,1\}^\ell$ into $x \in \{0,1\}^L$ using an admissible hash function. Then, we homomorphically derive $\mathbf{A}(x)$ as a GSW ciphertext $\mathbf{A}(x) = \mathbf{A} \cdot \mathbf{R}_x + (\prod_{i=1}^L \mu_{i,x[i]}) \cdot \mathbf{G}$, for some small-norm $\mathbf{R}_x \in \mathbb{Z}^{m \times m}$. By carefully choosing $\{\mu_{i,b}\}_{i \in [L], b \in \{0,1\}}$, the properties of admissible hash functions ensure that the product $\prod_{i=1}^L \mu_{i,x[i]}$ cancels out in all evaluation queries but evaluates to 1 on the challenge input $X^\star$.

In the next step of the proof, we move to a modified experiment where the random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is replaced by a lossy matrix $\mathbf{A}^\top = \bar{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E}$, where $\bar{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n' \times m})$, $\mathbf{C} \hookleftarrow U(\mathbb{Z}_q^{n' \times n})$ and $\mathbf{E} \in \mathbb{Z}^{m \times n}$ is a short Gaussian matrix. This modification has the consequence of turning $\lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rfloor_p$ into a lossy function of $\mathbf{s}$ on all inputs X for which $\prod_{i=1}^L \mu_{i,x[i]} = 0$. At the same time, the function remains injective whenever $\prod_{i=1}^L \mu_{i,x[i]} = 1$. Using the properties of admissible hash functions, we still have a noticeable probability that the function be lossy in all evaluation queries and injective in the challenge phase. Moreover, by using a small-norm secret $\mathbf{s} \in \mathbb{Z}^n$ and setting the ratio q/p large enough, we can actually make sure

that evaluation queries always reveal the same information (namely, the product $\mathbf{C}\cdot\mathbf{s}$) about $\mathbf{s}$. As long as we have $\prod_{i=1}^{L}\mu_{i,x^\star[i]}=1$ for the challenge input $X^\star$, the rounded value $\mathbf{z}^\star=\lfloor\mathbf{A}(x^\star)^\top\cdot\mathbf{s}\rfloor_p=\lfloor(\mathbf{A}\cdot\mathbf{R}_{x^\star}+\mathbf{G})^\top\cdot\mathbf{s}\rfloor_p$ is guaranteed to have a lot of entropy as an injective function of an unpredictable $\mathbf{s}$. At this point, we can extract statistically uniform bits from the source $\mathbf{z}^\star$. Since the latter depends on $x^\star$ (which can be correlated with the seed included in public parameters), we need an extractor that can operate on seed-dependent sources. Fortunately, deterministic extractors come in handy for this purpose.

3.4.2 The Centralized Construction

Let λ be a security parameter and let $\ell\in\Theta(\lambda)$, $L\in\Theta(\lambda)$. We use parameters consisting of prime moduli p and q such that $q=2^{\Theta(\lambda)}$, $p>2\log q$ and $q/p>2^{L+\lambda}\cdot r$, dimensions $n,m,k\in\mathsf{poly}(\lambda)$ such that $m\geq 2n\cdot\lceil\log q\rceil$ and $r=L\cdot m^2\cdot n\cdot\alpha q\cdot\beta$, where $\sigma=O(\sqrt{n})\cdot q^{n'/n}$, $\beta=\sigma\sqrt{n}$ and $\alpha\in(0,1)$. The integer $n'<n$ and $\alpha\in(0,1)$ are chosen such that the $\mathsf{LWE}_{q,m,n',\alpha}$ problem is hard.

We define the entropy bound $\bar{n}=\lfloor n\cdot\log\sigma-n'\cdot\log q\rfloor-1$ and choose k such that $k\cdot\log p=\bar{n}-2\cdot(\lambda+\log\ell+\log\bar{n})$.

The Gaussian parameter $\sigma>0$, will specify an interval $[-\beta,\beta]=[-\sigma\sqrt{n},\sigma\sqrt{n}]$ where the coordinates of the secret will be confined (with probability exponentially close to 1). We also rely on the following ingredients:

- A balanced admissible hash function $\mathsf{AHF}:\{0,1\}^\ell\to\{0,1\}^L$ (Def. 2.1).
- A family Π_λ of ξ-wise independent hash functions(Def. 2.2) $\pi:\mathbb{Z}_p^m\to\mathbb{Z}_p^k$ for $\xi=\bar{n}+\ell$. Let a random member π of Π_λ. For example, the function π can be a random polynomial $\pi(Z)\in\mathbb{F}_{p^m}[Z]$ of degree $\xi-1$ with outputs truncated to their k first coordinates.

The pseudo-random function family assumes the availability of public parameters

$$\mathsf{pp}:=\Big(q,\ \pi,\ \mathbf{U}_0,\ \{\mathbf{A}_{i,0},\mathbf{A}_{i,1}\ \in\mathbb{Z}_q^{n\times m}\}_{i=1}^{L},\ \mathsf{AHF},\ r,\ \sigma\Big),$$

where $\mathbf{U}_0\hookleftarrow U(\mathbb{Z}^{n\times m})$ and $\mathbf{A}_{i,0},\mathbf{A}_{i,1}\hookleftarrow U(\mathbb{Z}_q^{n\times m})$ for each $i\in[L]$.

Keygen(pp)**:** Given pp, sample a vector $\mathbf{s}\hookleftarrow D_{\mathbb{Z}^n,\sigma}$ so that $\|\mathbf{s}\|_\infty<\beta=\sigma\sqrt{n}$ with overwhelming probability. The secret key is $SK:=\mathbf{s}\in[-\beta,\beta]^n$.

Eval(pp, SK, X)**:** Given $SK=\mathbf{s}\in\mathbb{Z}^n$ and an input $X\in\{0,1\}^\ell$,

1. Compute $x=\mathsf{AHF}(X)\in\{0,1\}^L$ and parse it as $x=x_1\ldots x_L$.
2. Compute

$$\mathbf{z}=\Big\lfloor\big(\mathbf{A}(x)\big)^\top\cdot\mathbf{s}\Big\rceil_p\in\mathbb{Z}_p^m, \tag{3.1}$$

where

$$\mathbf{A}(x) = \mathbf{U}_0 + \mathbf{A}_{1,x_1} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{2,x_2} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{3,x_3} \cdot \cdots \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{L-1,x_{L-1}} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{L,x_L}\big)\big)\cdots\big),$$

and output $\mathbf{y} = \pi(\mathbf{z}) \in \mathbb{Z}_p^k$.

Unlike [BLMR13, BP14], our security proof requires the secret $\mathbf{s}$ to have small entries. Another difference is the presence of the uniformly random matrix $\mathbf{U}_0$ in the expression of $\mathbf{A}(x)$, which guarantees that the roundings from the proof behave in a predictable manner. Also, our PRF is not key-homomorphic as the output is $\mathbf{y} = \pi(\mathbf{z}) \in \mathbb{Z}_p^k$, thus the key-homomorphic property is ruined by the deterministic extractor. Fortunately, losing the key-homomorphic property does not prevent us from building a DPRF since the randomness extraction step is only applied to the result of combining t partial evaluations in the Combine algorithm.

Theorem 3.3. *The construction above is a secure PRF family under the* $\mathsf{LWE}_{q,m,n',\alpha}$ *assumption.*

The complete security proof for the above result can be found in [LST18]. We skip the details since the same ideas will be used in the decentralized case, for the proof of Theorem 3.5.

3.5 The DPRF Construction

We design the distributed PRF by doing secret sharing over the integers instead of a finite field. We use a LISS (section 2.6.2) inside the PRF construction of Section 3.4. As mentioned earlier, the latter is well-suited to our purposes because, in the security proof, the secret key is known to the reduction at any time. When the secret key $\mathbf{s}$ is shared using a LISS, the reduction is always able to consistently answer corruption queries because it has all shares at disposal.

In the construction, we rely on the specific LISS construction of Damgård and Thorbek [DT06], which is based on the Benaloh-Leichter secret sharing [BL88]. This particular LISS scheme is well-suited to our needs for several reasons. First, it has binary share generating matrices, which allows obtaining relatively short shares of $\mathbf{s} \in \mathbb{Z}^n$: in the security proof, this is necessary to ensure that the adversary always obtains the same information about uncorrupted shares in partial evaluation queries. Another advantage of the Benaloh-Leichter-based LISS is that its reconstruction coefficients live in $\{-1,0,1\}$, which avoids blowing up the homomorphism errors when partial evaluations are combined together. Finally, its sweeping vectors also have their coordinates in $\{-1,0,1\}$ (whereas they may be exponentially large in the number N of servers in the construction based on Cramer-Fehr [CF02]) and we precisely need sweeping vectors $\boldsymbol{\kappa}$ to be small in the proof of our Lemma 3.7.

When using the LISS scheme we do not rely on the result of Lemma 2.20, which ensures the privacy of the scheme. We share vectors sampled from Gaussian (instead of uniform) distributions and also depart from Lemma 2.20 in that the random coins $(\rho_2,\ldots,\rho_e)$ are not sampled from a wider distribution than the secret: the standard deviation of $(\rho_2,\ldots,\rho_e)$ will be the same as that of s. While this choice does not guarantee the LISS to be private in general, we will show that it suffices in our setting because we only need the secret to have sufficient min-entropy conditionally on the shares observed by the adversary. Aside from the distribution of secrets and random coins, we rely on the technique of Damgård and Thorbek [DT06] for building share generating matrices.

3.5.1 The Construction

Setup$(1^\lambda, 1^\ell, 1^t, 1^N)$**:** On input of a security parameter λ, a number of servers N, a threshold $t \in [1,N]$ and an input length $\ell \in \Theta(\lambda)$, set $d, e = O(N^{1+\sqrt{2}})$.

Next, do the following.

1. Choose prime moduli p, q and u such that $q = 2^{\Omega(\lambda)}$ that also satisfy $p/u > d\cdot m\cdot 2^{\lambda+L}$ and $q/p > m\cdot N\cdot d\cdot r\cdot 2^{\lambda+L}$, where dimensions $n, m, k \in \mathsf{poly}(\lambda)$ such that $m \geq 2n\cdot\lceil\log q\rceil$, and $r = L\cdot m^2\cdot n\alpha q\cdot\beta^*$ with $\beta^* := O(\beta\cdot\log N)$ where $[-\beta,\beta]^n$ is the space from where the secret key is sampled, and it is defined below.
2. Choose integer $n' \ll n$ and the Gaussian parameters $\alpha \in (0,1)$ and σ such that the $\mathsf{LWE}_{q,m,n',\alpha}$ problem is hard and define $\sigma := 2\sqrt{e}\cdot q^{n'/n}$.
   We also set the entropy bound $\bar{n} := n\cdot\log\left(\sigma/\sqrt{e}\right) - n'\cdot\log q$, which has $O(n)$ order of magnitude (by the choice of σ). We also choose integer k such that $k\cdot\log u = \bar{n} - 2\cdot\left(\lambda + \log\ell + \log\bar{n} + O(1)\right)$
3. Choose a balanced admissible hash function $\mathsf{AHF}: \{0,1\}^\ell \to \{0,1\}^L$, for a suitable $L \in \Theta(\lambda)$. Choose a family Π_λ of ξ-wise independent (Definition 2.2) hash functions $\pi : \mathbb{Z}_u^m \to \mathbb{Z}_u^k$, for $\xi = \bar{n} + \ell$, and sample a uniformly random function $\pi \hookleftarrow U(\Pi_\lambda)$.
4. Choose random matrices $\mathbf{U}_0 \hookleftarrow U(\mathbb{Z}_q^{n\times m})$ and $\mathbf{A}_{i,j} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, for each $i \in [L]$, $j \in \{0,1\}$.

Output

$$\mathsf{pp} := \left(q,\ p,\ u,\ \pi,\ \mathbf{U}_0,\ \{\mathbf{A}_{i,0},\mathbf{A}_{i,1}\ \in \mathbb{Z}_q^{n\times m}\}_{i=1}^L,\ \mathsf{AHF}, \sigma\right),$$

Keygen(pp)**:** Sample a Gaussian vector $\mathbf{s} \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ and return $\perp$ if $\mathbf{s} \notin [-\beta,\beta]^n$, where $\beta = \sigma\sqrt{n}$. Otherwise set $SK := \mathbf{s} \in \mathbb{Z}^n$.

Share(pp, SK)**:** Given pp and a key $SK = \mathbf{s} \in \mathbb{Z}^n$, generate a LISS of $\mathbf{s}$ as follows.

1. Using the Benaloh-Leichter-based LISS scheme (2.6.2), construct the matrix $M \in \{0,1\}^{d\times e}$ that computes the Boolean formula associated with the $T_{t,N}$ threshold function. By using [HHP06], we obtain a matrix $M \in \{0,1\}^{d\times e}$, so that each row of M contains $O(\log N)$ non-zero entries.
2. For each $k \in [n]$, generate a LISS of the k-th coordinate s_k of $\mathbf{s} \in \mathbb{Z}^n$. To this end, define a vector $\boldsymbol{\rho}_k = (s_k, \rho_{k,2}, \ldots, \rho_{k,e})^\top$, with Gaussian entries $\rho_{k,2}, \ldots, \rho_{k,e} \hookleftarrow D_{\mathbb{Z},\sigma}$, and compute
$$\mathbf{s}_k = (s_{k,1}, \ldots, s_{k,d})^\top = M \cdot \boldsymbol{\rho}_k \ \in \mathbb{Z}^d,$$
whose entries are smaller than $\|\mathbf{s}_k\|_\infty = \beta^* = O(\beta \cdot \log N)$.
3. Define the matrix $\mathbf{S} = [\mathbf{s}_1 \mid \ldots \mid \mathbf{s}_n] \in \mathbb{Z}^{d\times n}$. For each $j \in [N]$, define the share of server P_j to be the sub-matrix $\mathbf{S}_{I_j} = M_{I_j} \cdot [\boldsymbol{\rho}_1 \mid \ldots \mid \boldsymbol{\rho}_n] \in \mathbb{Z}^{d_j\times n}$, where $I_j = \psi^{-1}(j) \subset \{1, \ldots, d\}$ is the set of indexes such that P_j owns the sub-matrix $M_{I_j} \in \{0,1\}^{d_j\times e}$.

For each $j \in [N]$, the share $SK_j = \mathbf{S}_{I_j} \in \mathbb{Z}^{d_j\times n}$ is privately sent to P_j.

PEval(pp, SK_j, X)**:** Given SK_j as $\mathbf{S}_{I_j} = M_{I_j} \cdot [\boldsymbol{\rho}_1 \mid \ldots \mid \boldsymbol{\rho}_n] \in \mathbb{Z}^{d_j\times n}$ and an input $X \in \{0,1\}^\ell$,

1. Compute $x = \mathsf{AHF}(X) \in \{0,1\}^L$ and parse it as $x = x_1 \ldots x_L$.
2. Compute
$$\mathbf{Y}_j = \left\lfloor \left(\mathbf{A}(x)\right)^\top \cdot \mathbf{S}_{I_j}^\top \right\rceil_p \ \in \mathbb{Z}_p^{m\times d_j}, \tag{3.2}$$
where
$$\mathbf{A}(x) = \mathbf{U}_0 + \mathbf{A}_{1,x_1} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{2,x_2} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{3,x_3} \cdots \mathbf{G}^{-1}\big(\mathbf{A}_{L-1,x_{L-1}} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{L,x_L}\big)\big)\cdots\big)\big),$$
and output the partial evaluation $\mathbf{Y}_j \in \mathbb{Z}_p^{m\times d_j}$.

Eval(pp, SK, X)**:** Given $SK = \mathbf{s} \in \mathbb{Z}^n$ and an input $X \in \{0,1\}^\ell$,

1. Compute $x = \mathsf{AHF}(X) \in \{0,1\}^L$ and write it as $x = x_1 \ldots x_L$.
2. Compute
$$\mathbf{z} = \left\lfloor \left\lfloor \left(\mathbf{A}(x)\right)^\top \cdot \mathbf{s} \right\rceil_p \right\rfloor_u \ \in \mathbb{Z}_u^m,$$
where
$\mathbf{A}(x) = \mathbf{U}_0 + \mathbf{A}_{1,x_1} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{2,x_2} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{3,x_3} \cdots \mathbf{G}^{-1}\big(\mathbf{A}_{L-1,x_{L-1}} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{L,x_L}\big)\big)\cdots\big)\big)$,
and output $\mathbf{y} = \pi(\mathbf{z}) \in \mathbb{Z}_u^k$.

Combine$(\mathcal{S}, (\mathbf{Y}_{j_1}, \ldots, \mathbf{Y}_{j_t}))$**:** Let $\mathcal{S} = \{j_1, \ldots, j_t\}$ be a qualified set.

1. Determine the vector $\boldsymbol{\lambda}_{\mathcal{S}} \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$ such that $\boldsymbol{\lambda}_{\mathcal{S}}^{\top} \cdot M_{\mathcal{S}} = (1, 0, \ldots, 0)^{\top}$, where $M_{\mathcal{S}} \in \{0,1\}^{d_{\mathcal{S}} \times e}$ is the sub-matrix of M owned by the parties in $\mathcal{S}$ and $d_{\mathcal{S}} = \sum_{\kappa=1}^{t} d_{j_\kappa}$ with $d_{j_\kappa} = |\psi^{-1}(j_\kappa)|$ for all $\kappa \in [t]$. Then, parse $\boldsymbol{\lambda}_{\mathcal{S}}$ as $[\boldsymbol{\lambda}_{j_1}^{\top} \mid \ldots \mid \boldsymbol{\lambda}_{j_t}^{\top}]^{\top}$, where $\boldsymbol{\lambda}_{j_\kappa} \in \{-1, 0, 1\}^{d_{j_\kappa}}$ for all $\kappa \in [t]$.
2. Compute $\tilde{\mathbf{z}} = \sum_{\kappa=1}^{t} \mathbf{Y}_{j_\kappa} \cdot \boldsymbol{\lambda}_{j_\kappa} \in \mathbb{Z}_p^m$, which equals

$$\tilde{\mathbf{z}} \quad = \quad \left\lfloor \left(\mathbf{A}(x)\right)^{\top} \cdot \mathbf{s} \right\rceil_p + \mathbf{e}_z \in \mathbb{Z}_p^m,$$

   for some $\mathbf{e}_z \in \{-d_{\mathcal{S}}, \ldots, d_{\mathcal{S}}\}^m$.
3. Compute $\mathbf{z} = \lfloor \tilde{\mathbf{z}} \rceil_u \in \mathbb{Z}_u^m$, which equals

$$\mathbf{z} = \left\lfloor \left\lfloor \left(\mathbf{A}(x)\right)^{\top} \cdot \mathbf{s} \right\rceil_p \right\rfloor_u \in \mathbb{Z}_u^m$$

   with overwhelming probability. Finally, output $\mathbf{y} = \pi(\mathbf{z}) \in \mathbb{Z}_u^k$.

By setting $\sigma = 2\sqrt{e} \cdot q^{n'/n} = O(N^{\frac{1+\sqrt{2}}{2}}) \cdot q^{n'/n}$ as allowed by [HHP06], we have share units of magnitude $\beta^* = \Theta(\sigma\sqrt{n} \cdot \log N)$. Since $d = O(N^{1+\sqrt{2}})$, the average share size amounts to $\frac{d \cdot n \cdot \log \beta^*}{N} = O(n \cdot N^{\sqrt{2}} \cdot \log N) + O(N^{\sqrt{2}} \cdot n' \cdot \log q))$ bits.

Regarding the parameters, we need $\alpha q = \Omega(\sqrt{n'})$ for the LWE problem to be hard. We may set $n = \mathsf{poly}(\lambda)$ (for a small degree polynomial) and $q = 2^{\Omega(\lambda)}$.

We remark that our modulus q is exponential in the security parameter λ, but not in the number of server N. In contrast, the DPRF of [BLMR13] requires an exponential modulus in N incurred by the use of Shamir's secret sharing and the technique of clearing out the denominators [ABV+12].

3.5.2 Correctness and Security

We now show that the construction provides statistical consistency.

Lemma 3.4. *Let* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$ *and let a secret key* $SK = \mathbf{s} \leftarrow D_{\mathbb{Z}^n, \sigma}$, *which is shared as* $(SK_1, \ldots, SK_N) \leftarrow \mathsf{Share}(\mathsf{pp}, SK)$. *For any* t*-subset* $\mathcal{S} = \{j_1, \ldots, j_t\} \subset [N]$ *and input* $X \in \{0,1\}^\ell$, *if* $Y_{j_k} = \mathsf{PEval}(\mathsf{pp}, SK_{j_k}, X)$ *for all* $\kappa \in [t]$, *we have*

$$\mathsf{Combine}(\mathcal{S}, (Y_{j_1}, \ldots, Y_{j_t})) = \mathsf{Eval}(\mathsf{pp}, SK, X)$$

with probability exponentially close to 1.

Proof. Let $\boldsymbol{\lambda}_{\mathcal{S}} \in \{-1, 0, 1\}^{d_{\mathcal{S}}}$ such that $\boldsymbol{\lambda}_{\mathcal{S}}^{\top} \cdot M_{\mathcal{S}} = (1, 0, \ldots, 0) \in \mathbb{Z}^e$. If we parse $\boldsymbol{\lambda}_{\mathcal{S}}$ as $[\boldsymbol{\lambda}_{j_1}^{\top} \mid \ldots \mid \boldsymbol{\lambda}_{j_t}^{\top}]^{\top}$ we have

$$\mathbf{s} = [\boldsymbol{\rho}_1 \mid \cdots \mid \boldsymbol{\rho}_n]^{\top} \cdot M_{\mathcal{S}}^{\top} \cdot \boldsymbol{\lambda}_{\mathcal{S}} = \sum_{k=1}^{t} \mathbf{S}_{I_{j_k}}^{\top} \cdot \boldsymbol{\lambda}_{j_k}.$$

In turn, this implies

$$\lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rceil_p = \left\lfloor \sum_{k=1}^{t} \mathbf{A}(x)^\top \cdot \mathbf{S}_{I_{j_k}}^\top \cdot \boldsymbol{\lambda}_{j_k} \right\rceil_p = \sum_{k=1}^{t} \left\lfloor \mathbf{A}(x)^\top \cdot \mathbf{S}_{I_{j_k}}^\top \right\rceil_p \cdot \boldsymbol{\lambda}_{j_k} + \boldsymbol{e}, \tag{3.3}$$

where the last equality of (3.3) stems from fact that, for any two vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{Z}_q^m$, we have $\lfloor \boldsymbol{v}_1 + \boldsymbol{v}_2 \rceil_p = \lfloor \boldsymbol{v}_1 \rceil_p + \lfloor \boldsymbol{v}_2 \rceil_p + \boldsymbol{e}^+$, for some vector $\boldsymbol{e}^+ \in \{0,1\}^m$, and $\lfloor \boldsymbol{v}_1 - \boldsymbol{v}_2 \rceil_p = \lfloor \boldsymbol{v}_1 \rceil_p - \lfloor \boldsymbol{v}_2 \rceil_p + \boldsymbol{e}^-$, where $\boldsymbol{e}^- \in \{-1,0\}^m$. The error vector $\boldsymbol{e}$ of (3.3) thus lives in $\{-d_{\mathscr{S}}, \ldots, d_{\mathscr{S}}\}^m$. By the definition of $\mathbf{Y}_{j_k} = \lfloor \mathbf{A}(x)^\top \cdot \mathbf{S}_{I_{j_k}}^\top \rceil_p$, if we define $\tilde{\mathbf{z}} := \sum_{k=1}^{t} \mathbf{Y}_{j_k} \cdot \boldsymbol{\lambda}_{j_k}$ and $\boldsymbol{e}_z := -\boldsymbol{e} \in \{-d_{\mathscr{S}}, \ldots, d_{\mathscr{S}}\}^m$, we have

$$\tilde{\mathbf{z}} \quad = \quad \left\lfloor \big(\mathbf{A}(x)\big)^\top \cdot \mathbf{s} \right\rfloor_p + \mathbf{e}_z \in \mathbb{Z}_p^m.$$

Remember that $\mathbf{A}(x)^\top \cdot \mathbf{s}$ contains the term $\mathbf{U}_0^\top \cdot \mathbf{s}$, where the public matrix $\mathbf{U}_0$ is uniformly sampled over $\mathbb{Z}_q^{n \times m}$. By Lemma 2.11 it follows that $\mathbf{U}_0^\top \cdot \mathbf{s}$ is statistically close to $U(\mathbb{Z}_q^m)$. Thus the vector $\mathbf{A}(x)^\top \cdot \mathbf{s}$ is itself statistically close to $U(\mathbb{Z}_q^m)$. Hence, the vector $\lfloor \mathbf{A}(x)^\top \cdot \mathbf{s} \rceil_p$ is statistically close to $U(\mathbb{Z}_p^m)$ since the statistical distance between $\lfloor U(\mathbb{Z}_q^m) \rceil_p$ and $U(\mathbb{Z}_p^m)$ is at most $m \cdot (p/q)$. Therefore we can apply Lemma 2.10, which implies that

$$\left\lfloor \lfloor \mathbf{A}(x)^\top \cdot \boldsymbol{s} \rceil_p \right\rceil_u = \left\lfloor \lfloor \mathbf{A}(x)^\top \cdot \boldsymbol{s} \rceil_p + \boldsymbol{e}_z \right\rceil_u$$

except with probability $2^L \cdot m \cdot \frac{2d_{\mathscr{S}} \cdot u}{p} \leq 2^L \cdot m \cdot \frac{2d \cdot u}{p} \leq \cdot 2^{-\lambda}$.

This shows that the equality $\lfloor \tilde{\mathbf{z}} \rceil_u = \left\lfloor \lfloor \mathbf{A}(x)^\top \cdot \boldsymbol{s} \rceil_p \right\rceil_u$ holds with overwhelming probability if the vector $\tilde{\mathbf{z}} := \sum_{k=1}^{t} \mathbf{Y}_{j_k} \cdot \boldsymbol{\lambda}_{j_k}$ in the left-hand-side member is computed by the Combine algorithm and the right-hand-side member is the $\lfloor \tilde{\mathbf{z}} \rceil_u$ computed by Eval.

□

Theorem 3.5. *The construction above is an adaptively secure DPRF family under the* $\mathsf{LWE}_{q,m,n',\alpha}$ *assumption.*

Proof. The proof considers a sequence of hybrid games. In each game, we call W_i the event that $b' = b$.

Game$_0$: This is the experiment, as described by Definition 3.2. Namely, the challenger initially samples a secret Gaussian vector $SK = \mathbf{s} \hookleftarrow D_{\mathbb{Z}^n,\sigma}$, which is shared by computing

$$\mathbf{S}_{I_j} \quad = \quad M_{I_j} \cdot [\boldsymbol{\rho}_1 \,|\, \ldots \,|\, \boldsymbol{\rho}_n] = M_{I_j} \cdot \Gamma \in \mathbb{Z}^{d_j \times n} \qquad \forall\, j \in [N],$$

where

$$\Gamma = \big[\boldsymbol{\rho}_1 \,|\, \ldots \,|\, \boldsymbol{\rho}_n\big] = \left[\begin{array}{ccc} & \mathbf{s}^\top & \\ \hline \rho_{1,2} & \cdots & \rho_{n,2} \\ \vdots & \ddots & \vdots \\ \rho_{1,e} & \cdots & \rho_{n,e} \end{array}\right] \in \mathbb{Z}^{e \times n},$$

with $\rho_{k,v} \hookleftarrow D_{\mathbb{Z},\sigma}$ for all $(k,v) \in [1,n] \times [2,e]$. At each partial evaluation query $(j, X^{(i)}) \in [N] \times \{0,1\}^\ell$, the adversary $\mathscr{A}$ obtains

$$\mathbf{Y}_j = \left\lfloor (\mathbf{A}(x))^\top \cdot \mathbf{S}_{I_j}^\top \right\rceil_p \in \mathbb{Z}_p^{m \times d_j}. \tag{3.4}$$

In the challenge phase, the adversary chooses an input $X^\star \in \{0,1\}^\ell$. It obtains a random vector $\mathbf{y}^\star \hookleftarrow U(\mathbb{Z}_u^k)$ if the challenger's bit is $b = 0$. If $b = 1$, it obtains the real evaluation $\mathbf{y}^\star = \pi(\lfloor \tilde{\mathbf{z}}^\star \rceil_u) \in \mathbb{Z}_u^k$, where

$$\tilde{\mathbf{z}}^\star = \left\lfloor \left(\mathbf{A}(x^\star)\right)^\top \cdot \mathbf{s} \right\rceil_p \in \mathbb{Z}_p^m,$$

with $\mathbf{A}(x^\star) = \mathbf{U}_0 + \mathbf{A}_{1,x_1^\star} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{2,x_2^\star} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{3,x_3^\star} \cdots \mathbf{G}^{-1}\big(\mathbf{A}_{L-1,x_{L-1}^\star} \cdot \mathbf{G}^{-1}\big(\mathbf{A}_{L,x_L^\star}\big)\big)\cdots\big)\big)$.
At the end of the game, we define $\mathscr{C}^\star \subset [N]$ as in Definition 3.2, to be the set of servers that were corrupted by $\mathscr{A}$ or such that an evaluation query of the form $(i, X^\star)$ was made. By hypothesis, we have $|\mathscr{C}^\star| < t$. When the adversary halts, it outputs $\hat{b} \in \{0,1\}$ and the challenger defines $b' := \hat{b}$. The adversary's advantage is $\mathbf{Adv}_{\mathscr{A}}(\lambda) := |\Pr[W_0] - 1/2|$, where W_0 is event that $b' = b$.

Game$_1$: This game is identical to Game$_0$ with the following changes. First, the challenger runs $K \leftarrow \mathsf{AdmSmp}(1^\lambda, Q_{\max}, \delta)$ to generate a key $K \in \{0,1,\perp\}^L$ for a balanced admissible hash function $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$, with $\delta := \mathbf{Adv}_{\mathscr{A}}(\lambda)$ and $Q_{\max}$ is an upper bound on the number of queries that the adversary makes. When the adversary halts and outputs $\hat{b} \in \{0,1\}$, the challenger checks if the conditions

$$P_K(X^{(1)}) = \cdots = P_K(X^{(Q)}) = 1 \quad \wedge \quad P_K(X^\star) = 0 \tag{3.5}$$

are satisfied, where $X^\star$ is the challenge input and $X^{(1)}, \ldots, X^{(Q)}$ are the adversarial queries. If these conditions do not hold, the challenger ignores $\mathscr{A}$'s output $\hat{b} \in \{0,1\}$ and overwrites it with a random bit $b'' \hookleftarrow \{0,1\}$ to define $b' = b''$. If conditions (3.5) are satisfied, the challenger sets $b' = \hat{b}$. By Lemma 2.17, we have

$$\begin{aligned} |\Pr[W_1] - 1/2| &= |\Pr[b' = b] - 1/2| \\ &\geq \gamma_{\min} \cdot \mathbf{Adv}_{\mathscr{A}}(\lambda) - \frac{1}{2} \cdot (\gamma_{\max} - \gamma_{\min}) = \tau(\lambda), \end{aligned}$$

where $\tau(\lambda)$ is a noticeable function.

Game$_2$: In this game, we modify the generation of pp in the following way. Initially, the challenger samples a uniformly random matrix $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n \times m})$. Next, for each $i \in [L]$, it samples $\mathbf{R}_{i,0}, \mathbf{R}_{i,1} \hookleftarrow U(\{-1,1\})^{m \times m}$ and defines the matrices $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^L$ as follows for all $i \in [L]$ and $j \in \{0,1\}$:

$$\mathbf{A}_{i,j} := \begin{cases} \mathbf{A} \cdot \mathbf{R}_{i,j} & \text{if } (j \neq K_i) \wedge (K_i \neq \perp) \\ \mathbf{A} \cdot \mathbf{R}_{i,j} + \mathbf{G} & \text{if } (j = K_i) \vee (K_i = \perp) \end{cases} \tag{3.6}$$

It also defines $\mathbf{U}_0 = \mathbf{A} \cdot \mathbf{V}_0$ for $\mathbf{V}_0 \hookleftarrow U(\{-1,1\}^{m \times m})$. Since $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ was chosen uniformly, the Leftover Hash Lemma 2.18 ensures that $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^{L}$, $\mathbf{U}_0$ are all statistically independent and uniformly distributed over $\mathbb{Z}_q^{n \times m}$. It follows that $|\Pr[W_2] - \Pr[W_1]| \leq L \cdot 2^{-\lambda}$ since the distribution of pp is statistically unchanged.

We note that, at each query X, we can view $\mathbf{A}(x)$ as a shifted by $\mathbf{U}_0$, GSW encryption

$$\mathbf{A}(x) = \mathbf{U}_0 + \mathbf{A} \cdot \mathbf{R}_x + (\prod_{i=1}^{n} \mu_i) \cdot \mathbf{G},$$

which is also equal to:

$$\mathbf{A}(x) = \mathbf{A} \cdot \big(\mathbf{V}_0 + \mathbf{R}_x\big) + (\prod_{i=1}^{n} \mu_i) \cdot \mathbf{G},$$

for some small norm $\mathbf{R}_x \in \mathbb{Z}^{m \times m}$, where

$$\mu_i := \begin{cases} 0 & \text{if } (\mathsf{AHF}(X)_i \neq K_i) \wedge (K_i \neq \perp) \\ 1 & \text{if } (\mathsf{AHF}(X)_i = K_i) \vee (K_i = \perp) \end{cases}$$

If conditions (3.5) are satisfied, at each query $X^{(i)}$, the admissible hash function ensures that $x^{(i)} = \mathsf{AHF}(X^{(i)})$ satisfies

$$\mathbf{A}(x^{(i)}) = \mathbf{A} \cdot \big(\mathbf{V}_0 + \mathbf{R}_{x^{(i)}}\big), \tag{3.7}$$

for some small norm $\mathbf{R}_{x^{(i)}} \in \mathbb{Z}^{m \times m}$. Moreover, the admissible hash function maps the challenge input $X^\star$ to an L-bit string $x^\star = \mathsf{AHF}(X^\star)$ such that

$$\mathbf{A}(x^\star) = \mathbf{A} \cdot \big(\mathbf{V}_0 + \mathbf{R}_{x^\star}\big) + \mathbf{G}, \tag{3.8}$$

for some small norm $\mathbf{R}_{x^\star} \in \mathbb{Z}^{m \times m}$

Game$_3$: In this game, we modify the distribution of pp and replace the uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ by a lossy matrix such that

$$\mathbf{A}^\top = \bar{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E} \in \mathbb{Z}_q^{m \times n}, \tag{3.9}$$

where $\bar{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n' \times m})$, $\mathbf{C} \hookleftarrow U(\mathbb{Z}_q^{n' \times n})$ and $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m \times n}, \alpha q}$, for n' significantly smaller than n. The matrix in (3.9) is thus "computationally close" to a matrix $\bar{\mathbf{A}}^\top \cdot \mathbf{C}$ of much lower rank than n. Under the LWE assumption in dimension n', this change should not significantly alter $\mathscr{A}$'s behavior and a straightforward reduction $\mathscr{B}$ shows that $|\Pr[W_3] - \Pr[W_2]| \leq n \cdot \mathbf{Adv}_{\mathscr{B}}^{\mathsf{LWE}_{q,m,n',\alpha}}(\lambda)$, where the factor n comes from the use of an LWE assumption with n secrets.

The modification introduced in Game_3 has the following consequence. Assuming that conditions (3.5) are satisfied, for each partial evaluation query $X^{(i)}$ such that $X^{(i)} \neq X^\star$, the response is of the form

$$\begin{aligned}\mathbf{Y}_j &= \left\lfloor \left(\mathbf{A}\cdot\mathbf{V}_0 + \mathbf{A}\cdot\mathbf{R}_{x^{(i)}}\right)^\top \cdot \mathbf{S}_{I_j}^\top \right\rfloor_p \\ &= \left\lfloor \left(\mathbf{V}_0^\top + \mathbf{R}_{x^{(i)}}^\top\right)\cdot\left(\bar{\mathbf{A}}^\top\cdot\mathbf{C} + \cdot\mathbf{E}\right)\cdot\mathbf{S}_{I_j}^\top \right\rfloor_p \qquad \forall \theta \in [d_j].\end{aligned}$$

Game_4: In this game, we modify the evaluation oracle and the challenge oracle by introducing a bad event. We define BAD to be the event that the adversary makes a partial evaluation query (j, X) or a challenge query $X^\star$ such that the AHF-encoded inputs $x = \mathsf{AHF}(X)$ and $x^\star = \mathsf{AHF}(X^\star) \in \{0,1\}^L$ corresponding to matrices $\mathbf{A}(x) = \mathbf{A}\cdot\mathbf{R}_x$, and $\mathbf{A}(x^\star) = \mathbf{A}\cdot\mathbf{R}_{x^\star} + \mathbf{G}$ respectively, for some small-norm matrices $\mathbf{R}_x$ and $\mathbf{R}_{x^\star} \in \mathbb{Z}^{m\times m}$, satisfy either

$$\mathbf{Y}_j = \left\lfloor \left(\mathbf{A}\cdot\mathbf{V}_0 + \mathbf{A}\cdot\mathbf{R}_x\right)^\top \cdot \mathbf{S}_{I_j}^\top \right\rfloor_p \neq \left\lfloor \left(\mathbf{V}_0^\top + \mathbf{R}_x^\top\right)\cdot\bar{\mathbf{A}}^\top\cdot\mathbf{C}\cdot\mathbf{S}_{I_j}^\top \right\rfloor_p . \tag{3.10}$$

or

$$\tilde{\mathbf{z}}^\star = \left\lfloor \left(\mathbf{A}\cdot\mathbf{V}_0 + \mathbf{A}\cdot\mathbf{R}_{x^\star} + \mathbf{G}\right)^\top \cdot \mathbf{s} \right\rfloor_p \neq \left\lfloor \left(\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top\right)\cdot\bar{\mathbf{A}}^\top\cdot\mathbf{C}\cdot\mathbf{s} + \mathbf{G}^\top\cdot\mathbf{s} \right\rfloor_p . \tag{3.11}$$

Note that the challenger can detect this event since it knows $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n'\times m}$, $\mathbf{C} \in \mathbb{Z}_q^{n'\times n}$ and $\mathbf{E} \in \mathbb{Z}^{m\times n}$ satisfying (3.9). If BAD occurs, the challenger overwrites $\mathscr{A}$'s output $\hat{b}$ with a random bit $b'' \hookleftarrow \{0,1\}$ and sets $b' = b''$ (otherwise, it sets $b' = \hat{b}$ as before). Lemma 3.6 shows that we have the inequality $|\Pr[W_4] - \Pr[W_3]| \leq \Pr[\mathsf{BAD}] \leq 2^{-\Omega(\lambda)}$.

We note that, if BAD does not occur, we have

$$\left\lfloor \left(\mathbf{A}\cdot\mathbf{V}_0 + \mathbf{A}\cdot\mathbf{R}_{x^{(i)}}\right)^\top \cdot \mathbf{S}_{I_j}^\top \right\rfloor_p = \left\lfloor \left(\mathbf{V}_0^\top + \mathbf{R}_{x^{(i)}}^\top\right)\cdot\bar{\mathbf{A}}^\top\cdot\mathbf{C}\cdot\mathbf{S}_{I_j}^\top \right\rfloor_p \tag{3.12}$$

at each query $(j, X^{(i)})$ for which $X^{(i)} \neq X^\star$. We note that the right-hand-side member of (3.12) is fully determined by $\left(\mathbf{V}_0^\top + \mathbf{R}_{x^{(i)}}^\top\right)\cdot\bar{\mathbf{A}}^\top$ and the product $\mathbf{C}\cdot\mathbf{S}_{I_j}^\top \in \mathbb{Z}_q^{n'\times d_j}$. Conversely, the right-hand-side member of (3.12) uniquely determines $\mathbf{C}\cdot\mathbf{S}_{I_j}^\top$ with high probability: observe that $\left(\mathbf{V}_0^\top + \mathbf{R}_{x^{(i)}}^\top\right)\cdot\bar{\mathbf{A}}^\top$ is statistically uniform over $\mathbb{Z}_q^{m\times n'}$, so by Corollary 2.3.1, the quantity $\lfloor(\mathbf{V}_0^\top + \mathbf{R}_{x^{(i)}}^\top)\cdot\bar{\mathbf{A}}^\top\cdot(\mathbf{C}\cdot\mathbf{S}_{I_j}^\top)\rfloor_p$ is an injective function of $\mathbf{C}\cdot\mathbf{S}_{I_j}^\top \bmod q$.

This means that partial evaluation queries $(j, X^{(i)})$ such that $X^{(i)} \neq X^\star$ always reveal the same information (namely, $\mathbf{C}\cdot\mathbf{S}_{I_j}^\top \in \mathbb{Z}_q^{n'\times d_j}$) about $\mathbf{S}_{I_j}^\top \in \mathbb{Z}^{n\times d_j}$. It comes that partial evaluation queries information-theoretically reveal $\mathbf{C}\cdot\mathbf{S}^\top \bmod q$, in particular they reveal $\mathbf{C}\cdot\mathbf{s} \bmod q$. But we will show that $\mathbf{s}$ still retains high entropy in $\mathscr{A}$'s view.

Game$_5$: In this game, we modify the challenge value for which, if $b = 1$, the adversary is given a random $\mathbf{y}^\star \hookleftarrow U(\mathbb{Z}_u^k)$. Clearly, we have $\Pr[W_5] = 1/2$ since the distribution of the challenge value does not depend on $b \in \{0,1\}$. Moreover, we will show that $|\Pr[W_5] - \Pr[W_4]| \leq 2^{-\Omega(\lambda)}$.

Indeed, we claim that, conditionally on $\mathscr{A}$'s view, the vector $\mathbf{y}^\star$ is already statistically uniform over $\mathbb{Z}_u^k$ in Game$_4$. Indeed, the source $\lfloor \tilde{\mathbf{z}}^\star \rceil_u$ depends on an injective function $\mathbf{G}^\top \cdot \mathbf{s} \in \mathbb{Z}_q^m$ of the vector $\mathbf{s}$. In Lemma 3.7, we show that this vector has high min-entropy if BAD does not occur.

We observe that the source $\lfloor \tilde{\mathbf{z}}^\star \rceil_u$ can be written

$$\begin{aligned}
\lfloor \tilde{\mathbf{z}}^\star \rceil_u &= \left\lfloor \left\lfloor \left(\mathbf{A}\cdot\mathbf{V}_0 + \mathbf{A}\cdot\mathbf{R}_{x^\star} + \mathbf{G}\right)^\top \cdot \mathbf{s} \right\rceil_p \right\rceil_u && (3.13)\\
&= \left\lfloor \left(\mathbf{A}\cdot\mathbf{V}_0 + \mathbf{A}\cdot\mathbf{R}_{x^\star} + \mathbf{G}\right)^\top \cdot \mathbf{s} \right\rceil_u + \mathbf{e}_{s,x,u} && \text{with } \mathbf{e}_{s,x,u} \in \{-1,0\}^m\\
&= \lfloor (\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top)\cdot\mathbf{A}^\top \cdot \mathbf{s} \rceil_u + \lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}_{s,x,u} + \mathbf{e}_{s,x}, \text{ with } \mathbf{e}_{s,x} \in \{0,1\}^m\\
&= \lfloor (\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top)\cdot\mathbf{A}^\top \cdot \mathbf{s} \rceil_u + \lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}'_{s,x}, \quad \text{with } \mathbf{e}'_{s,x} \in \{-1,0,1\}^m. && (3.14)
\end{aligned}$$

Recall that the equality $\lfloor (\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top)\cdot\mathbf{A}^\top \cdot \mathbf{s} \rceil_u = \lfloor (\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top)\cdot\bar{\mathbf{A}}^\top \cdot \mathbf{C}\cdot\mathbf{s} \rceil_u$, corresponding to the challenge query, holds as long as the event BAD does not occur.

This implies $H_\infty\big(\lfloor (\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top)\cdot\mathbf{A}^\top \cdot \mathbf{s} \rceil_u \mid \mathbf{C}\cdot\mathbf{s}\big) = 0$ as long as BAD does not occur. In the expression of $\tilde{\mathbf{z}}^\star$ in (3.14), we also remark that $\lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}'_{s,x}$ is an injective function of $\mathbf{s} \in \mathbb{Z}^n$. To see this, observe that

$$\lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}'_{s,x} = (u/q)\cdot\mathbf{G}^\top \cdot \mathbf{s} - \mathbf{t}_{s,x} + \mathbf{e}'_{s,x}$$

for some $\mathbf{t}_{s,x} \in (0,1)^m$, so that

$$(q/u)\cdot(\lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}'_{s,x}) = \mathbf{G}^\top \cdot \mathbf{s} + \mathbf{e}''_{s,x} \tag{3.15}$$

for some $\mathbf{e}''_{s,x} \in (-q/u, q/u)^m$. The vector $\mathbf{s}$ is thus uniquely determined by (3.15), using the public trapdoor of $\mathbf{G}$, when $u > 2\log q$.

Consider the entropy of $z^\star := \lfloor \tilde{\mathbf{z}}^\star \rceil_u$ conditionally on $\mathscr{A}$'s view. We have

$$\begin{aligned}
H_\infty\big(\lfloor \tilde{\mathbf{z}}^\star \rceil_u \quad &| \quad \mathbf{C}\cdot\Gamma^\top,\ \{\mathbf{S}_{I_j}\}_{j\in\mathscr{C}^\star}\big)\\
&= H_\infty\big(\lfloor (\mathbf{V}_0^\top + \mathbf{R}_{x^\star}^\top)\cdot\mathbf{A}^\top \cdot \mathbf{s} \rceil_u + \lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}'_{s,x} \mid \mathbf{C}\cdot\Gamma^\top,\ \{\mathbf{S}_{I_j}\}_{j\in\mathscr{C}^\star}\big)\\
&= H_\infty\big(\lfloor \mathbf{G}^\top \cdot \mathbf{s} \rceil_u + \mathbf{e}'_{s,x} \mid \mathbf{C}\cdot\Gamma^\top,\ \{\mathbf{S}_{I_j}\}_{j\in\mathscr{C}^\star}\big)\\
&= H_\infty\big(\mathbf{s} \mid \mathbf{C}\cdot\Gamma^\top,\ \{\mathbf{S}_{I_j}\}_{j\in\mathscr{C}^\star}\big) \geq n\cdot\log\sigma - \frac{n}{2}\cdot\log e - n'\cdot\log q - 1.
\end{aligned}$$

Here, the last inequality is given by Lemma 3.7. The second equality follows from the fact that, for any random variables X, Y, Z defined over an additive group, we have $H_\infty(Y + Z \mid X) = H_\infty(Z|X)$ if $H_\infty(Y|X) = 0$.

In order to extract statistically random bits from $\mathbf{z}^\star$, we must take into account that it possibly depends on $x^\star$ which may depend on the public parameters. As long as $P_K(X^\star) = 0$, the source $\mathbf{z}^\star$ is taken from a distribution determined by the challenge input $X^\star \in \{0,1\}^\ell$ within a collection of less than 2^ℓ distributions (namely, those inputs X for which $P_K(X) = 0$), which all have min-entropy

$\bar{n} \geq n\log\sigma - \frac{n}{2}\cdot\log e - n'\log q - 1$. By applying Lemma 2.19 with $\epsilon = 2^{-\lambda}$ for a collection $\mathscr{X}$ of at most $M = 2^{\ell}$ distributions, we obtain that the distribution of $\pi(\mathbf{z}^{\star})$ is $2^{-\Omega(\lambda)}$-close to the uniform distribution over $\mathbb{Z}_u^k$. □

Lemma 3.6. *Assume that $q/p > 2^{L+\lambda}\cdot r\cdot m\cdot N\cdot d$, where $r = L\cdot m^2\cdot n\cdot\alpha q\cdot\beta^*$ with $\beta^* = O(\beta\cdot\log N)$. Then, we have the inequality*

$$|\Pr[W_4] - \Pr[W_3]| \leq \Pr[\mathsf{BAD}] \leq 2^{-\Omega(\lambda)}.$$

Proof. For each query $x = x_1\ldots x_L$, the matrix $\mathbf{R}_x \in \mathbb{Z}^{m\times m}$ is of the form

$$\begin{aligned}\mathbf{R}_x \;=\;& \mathbf{R}_{1,x_1}\cdot\mathbf{G}^{-1}\left(\mathbf{A}_{2,x_2}\cdot\mathbf{G}^{-1}\left(\mathbf{A}_{3,x_3}\cdot\left(\cdots\mathbf{G}^{-1}\left(\mathbf{A}_{L,x_L}\right)\right)\right)\right)\\ &+x_1\cdot\mathbf{R}_2\cdot\mathbf{G}^{-1}\left(\mathbf{A}_{3,x_3}\cdot\left(\cdots\mathbf{G}^{-1}\left(\mathbf{A}_{L,x_L}\right)\right)\right)\\ &+x_1\cdot x_2\cdot\mathbf{R}_3\cdot\mathbf{G}^{-1}\left(\mathbf{A}_{4,x_4}\cdot\left(\cdots\mathbf{G}^{-1}\left(\mathbf{A}_{L,x_L}\right)\right)\right)\\ &\vdots\\ &+x_1\cdot x_2\cdots x_{L-2}\cdot\mathbf{R}_{L-1}\cdot\mathbf{G}^{-1}\left(\mathbf{A}_{L,x_L}\right)\\ &+x_1\cdot x_2\cdots x_{L-1}\cdot\mathbf{R}_L\end{aligned}$$

Since $\mathbf{R}_x$ is the sum of L products of at most two binary matrices, the following bound can be established: $\|\mathbf{R}_x^{\top}\|_{\infty} \leq (L-1)\cdot m^2 + m$. Moreover, for every column $\bar{\mathbf{s}}_{j,\theta} \in \mathbb{Z}^n$, $\theta \in [d_j]$ of the matrix $\mathbf{S}_{I_j}^{\top} \in \mathbb{Z}^{n\times d_j}$, we have $\|\bar{\mathbf{s}}_{j,\theta}\|_{\infty} \leq \beta^*$. We also have $\|\mathbf{V}_0^{\top}\|_{\infty} \leq m$ and

$$\|\mathbf{E}\|_{\infty} = \max_{i\in[m]}\left(\sum_{j=1}^{n}|e_{i,j}|\right) \leq \sqrt{n}\max_{i\in[m]}\|\mathbf{e}_i\| \leq n\cdot\alpha q$$

so we can conclude that for every column $\bar{\mathbf{s}}_{j,\theta}$ of $\mathbf{S}_{I_j}^{\top}$ we have:

$$\|\left(\mathbf{V}_0^{\top}+\mathbf{R}_x^{\top}\right)\cdot\mathbf{E}\cdot\bar{\mathbf{s}}_{j,\theta}\|_{\infty} \leq \left(\|\mathbf{V}_0^{\top}\|_{\infty}+\|\mathbf{R}_x^{\top}\|_{\infty}\right)\cdot\|\mathbf{E}\|_{\infty}\cdot\|\bar{\mathbf{s}}_{j,\theta}\|_{\infty} \leq \left((L-1)m^2+2m\right)\cdot n\cdot\alpha q\cdot\beta^*,$$

which is smaller than r, for $r = L\cdot m^2\cdot n\cdot\alpha q\cdot\beta^*$.

By the Leftover Hash Lemma 2.18, we know that $\mathbf{V}_0^{\top}\cdot\bar{\mathbf{A}}^{\top}$ is statistically close to the uniform over $\mathbb{Z}_q^{m\times n}$. Thus by Lemma 2.11, we conclude that $(\mathbf{V}_0^{\top}\cdot\bar{\mathbf{A}}^{\top})\cdot(\mathbf{C}\cdot\bar{\mathbf{s}}_{j,\theta})$ is close to the uniform distribution over $\mathbb{Z}_q^m$. This implies that $(\mathbf{V}_0^{\top}+\mathbf{R}_x^{\top})\cdot\bar{\mathbf{A}}^{\top}\cdot\mathbf{C}\cdot\bar{\mathbf{s}}_{j,\theta}$ is statistically close to the uniform distribution over $\mathbb{Z}_q^m$.

So we can apply Lemma 2.10 to establish the following bound on $\Pr[\mathsf{bad}_{x,j,\theta}]$:

$$\Pr\left[\left\lfloor\left(\mathbf{A}\cdot\mathbf{V}_0+\mathbf{A}\cdot\mathbf{R}_x\right)^{\top}\cdot\bar{\mathbf{s}}_{j,\theta}\right\rceil_p \neq \left\lfloor\left(\mathbf{V}_0^{\top}+\mathbf{R}_x^{\top}\right)\cdot\bar{\mathbf{A}}^{\top}\cdot\mathbf{C}\cdot\bar{\mathbf{s}}_{j,\theta}\right\rceil_p\right] \leq m\cdot\frac{2rp}{q}. \tag{3.16}$$

where $r = L\cdot m^2\cdot n\cdot\alpha q\cdot\beta^*$, with $\beta^* = O(\beta\cdot\log N)$.

An almost identical argument can be used to establish a similar bound for the challenge query:

$$\Pr\left[\left\lfloor\left(\mathbf{A}\cdot\mathbf{V}_0+\mathbf{A}\cdot\mathbf{R}_{x^\star}+\mathbf{G}\right)^\top\cdot\mathbf{s}\right\rceil_p \neq \left\lfloor\left(\mathbf{V}_0^\top+\mathbf{R}_{x^\star}^\top\right)\cdot\bar{\mathbf{A}}^\top\cdot\mathbf{C}\cdot\mathbf{s}+\mathbf{G}^\top\cdot\mathbf{s}\right\rceil_p\right] \leq m\cdot\frac{2rp}{q} \quad (3.17)$$

By taking a union bound over all possible inputs $x\in\{0,1\}^L$ and all possible pairs $(j,\theta)\in[N]\times[d_j]$, we obtain the upper bound $\Pr[\mathsf{BAD}]\leq 2^L\cdot m\cdot N\cdot d\cdot\frac{2rp}{q}$.

Since Game_3 and Game_4 are identical if BAD does not occur, we finally obtain

$$|\Pr[W_4]-\Pr[W_3]|\leq\Pr[\mathsf{BAD}]\leq 2^L\cdot m\cdot N\cdot d\cdot\frac{2rp}{q}\leq 2^{-\lambda},$$

which completes the proof of the lemma. □

Lemma 3.7. *In* Game_4*, the min-entropy of* $\mathbf{s}$ *conditionally on* $\mathscr{A}$*'s view is at least* $n\cdot\log\sigma-\frac{n}{2}\cdot\log e-n'\cdot\log q-\frac{n}{2^n}$.

Proof. Let us assume that BAD does not occur in Game_4 since, if it does, the challenger replaces the adversary's output with a random bit, in which case both games have the same outcome. We show that, assuming ¬BAD, the shared secret vector $\mathbf{s}$ retains high min-entropy conditionally on the adversary's view.

Let us first recap what the adversary can see in Game_4. For each partial evaluation query $(j,X^\star)$, the response $\left\lfloor\left(\mathbf{A}\cdot\mathbf{V}_0+\mathbf{A}\cdot\mathbf{R}_{x^\star}+\mathbf{G}\right)^\top\cdot\mathbf{S}_{I_j}^\top\right\rceil_p$ consists of non-lossy functions of $\mathbf{S}_{I_j}^\top\in\mathbb{Z}^{n\times d_j}$. We thus consider partial evaluation queries of the form $(j,X^\star)$ as if they were corruption queries and assume that they information-theoretically reveal $\mathbf{S}_{I_j}$ (i.e. we assume $\mathscr{C}^\star$ from Definition 3.2 consists only of corrupted indices). As for uncorrupted shares $\{\mathbf{S}_{I_j}\}_{j\in[N]\setminus\mathscr{C}^\star}$, partial evaluation queries $(j,X^{(i)})$ for which $X^{(i)}\neq X^\star$ only reveal the information $\{\mathbf{C}\cdot\mathbf{S}_{I_j}^\top\}_{j\in[N]\setminus\mathscr{C}^\star}$. More precisely, those partial evaluations $\{\mathbf{Y}_j\}_{j\in[N]\setminus\mathscr{C}^\star}$ can be written

$$\mathbf{Y}_j \quad=\quad \left\lfloor(\mathbf{A}(x^{(i)}))^\top\cdot\mathbf{S}_{I_j}^\top\right\rceil_p=\left\lfloor\left(\mathbf{V}_0^\top+\mathbf{R}_{x^{(i)}}^\top\right)\cdot\bar{\mathbf{A}}^\top\cdot\mathbf{C}\cdot\mathbf{S}_{I_j}^\top\right\rceil_p \quad (3.18)$$

where

$$\mathbf{S}_{I_j}^\top=\begin{bmatrix}\boldsymbol{\rho}_1^\top\\ \vdots\\ \boldsymbol{\rho}_n^\top\end{bmatrix}\cdot M_{I_j}^\top\ \in\mathbb{Z}^{n\times d_j}$$

is a product of $M_{I_j}^\top$ with the matrix $[\boldsymbol{\rho}_1\,|\,\ldots\,|\,\boldsymbol{\rho}_n]^\top\in\mathbb{Z}^{n\times e}$ whose first column is the secret $SK=\mathbf{s}\in\mathbb{Z}^n$. Hence, the information revealed by (3.18) for $j\in[N]\setminus\mathscr{C}^\star$ is only a lossy function $\mathbf{C}\cdot\mathbf{S}_{I_j}^\top$ of the share $\mathbf{S}_{I_j}$: namely,

$$\begin{aligned}\mathbf{C}\cdot\begin{bmatrix}\boldsymbol{\rho}_1^\top\\ \vdots\\ \boldsymbol{\rho}_n^\top\end{bmatrix}\cdot M_{I_j}^\top &= \left[\ \mathbf{C}\cdot\mathbf{s}\ \middle|\ \mathbf{C}\cdot\begin{pmatrix}\rho_{1,2}\\ \vdots\\ \rho_{n,2}\end{pmatrix}\ \middle|\ \ldots\ \middle|\ \mathbf{C}\cdot\begin{pmatrix}\rho_{1,e}\\ \vdots\\ \rho_{n,e}\end{pmatrix}\ \right]\cdot M_{I_j}^\top, \quad (3.19)\\ &= \mathbf{C}\cdot\Gamma^\top\cdot M_{I_j}^\top,\end{aligned}$$

where

$$\Gamma = \left[\boldsymbol{\rho}_1 \mid \ldots \mid \boldsymbol{\rho}_n\right] = \left[\begin{array}{ccc} & \mathbf{s}^\top & \\ \hline \rho_{1,2} & \cdots & \rho_{n,2} \\ \vdots & \ddots & \vdots \\ \rho_{1,e} & \cdots & \rho_{n,e} \end{array}\right] \in \mathbb{Z}^{e\times n}$$

is the matrix of Gaussian entries which is used to compute secret key shares

$$\mathbf{S}_{I_j} = M_{I_j} \cdot \Gamma \qquad\qquad \forall\, j \in [N].$$

The information revealed by exposed shares $\{\mathbf{S}_{I_j}\}_{j\in\mathscr{C}^\star}$ can thus be written

$$\mathbf{S}_{I_j} = [\mathbf{s}_{I_j,1} \mid \ldots \mid \mathbf{s}_{I_j,n}] = M_{I_j} \cdot \Gamma \ \in \mathbb{Z}^{d_j\times n} \qquad\qquad \forall\, j \in \mathscr{C}^\star. \tag{3.20}$$

At this stage, we see that proving the following fact on distributions is sufficient to complete the proof of the lemma.

Claim. *Let $M_{\mathscr{C}^\star}$ to be the sub-matrix of M obtained by stacking up the rows assigned to corrupted parties $j \in \mathscr{C}^\star$. Conditionally on*

$$\big(\mathbf{C},\ \mathbf{C}\cdot\Gamma^\top\cdot M^\top,\ M_{\mathscr{C}^\star},\ M_{\mathscr{C}^\star}\cdot\Gamma\big), \tag{3.21}$$

the vector $\mathbf{s}^\top = (1,0,\ldots,0)^\top\cdot\Gamma$ has min-entropy $\geq n\cdot\log\sigma - \frac{n}{2}\cdot\log e - n'\cdot\log q - \frac{n}{2^n}$.

To prove this statement, we apply arguments inspired from [ALS16, Lemma 1]. First, we observe that conditioning on (3.21) is the same as conditioning on $\big(\mathbf{C}, \mathbf{C}\cdot\Gamma^\top\cdot M_{[N]\setminus\mathscr{C}^\star}^\top, M_{\mathscr{C}^\star}, M_{\mathscr{C}^\star}\cdot\Gamma\big)$ since $M_{\mathscr{C}^\star}\cdot\Gamma$ and $\mathbf{C}$ are given. In fact, it is sufficient to prove the result when conditioning on

$$\big(\mathbf{C}, \mathbf{C}\cdot\Gamma^\top, M_{\mathscr{C}^\star}, M_{\mathscr{C}^\star}\cdot\Gamma\big),$$

as $\mathbf{C}\cdot\Gamma^\top\cdot M_{[N]\setminus\mathscr{C}^\star}^\top$ is computable from $\mathbf{C}\cdot\Gamma^\top$. By the definition of an Integer Span Program, we know that there exists a sweeping vector $\boldsymbol{\kappa}\in\mathbb{Z}^e$ whose first coordinate is $\kappa_1 = 1$ and such that $M_{\mathscr{C}^\star}\cdot\boldsymbol{\kappa} = \mathbf{0}$. The rows of $M_{\mathscr{C}^\star}$ thus live in the lattice $\mathscr{L}_{\mathscr{C}^\star} = \{\boldsymbol{m}\in\mathbb{Z}^e : \langle\boldsymbol{m},\boldsymbol{\kappa}\rangle = 0\}$. Hence, if we define a matrix $L_{\mathscr{C}^\star}\in\mathbb{Z}^{(e-1)\times e}$ whose rows form a basis of $\mathscr{L}_{\mathscr{C}^\star}$, we may prove the min-entropy lower bound conditioned on

$$\big(\mathbf{C}, \mathbf{C}\cdot\Gamma^\top, L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma\big).$$

This is because $L_{\mathscr{C}^\star}\cdot\Gamma$ provides at least as much information as $M_{\mathscr{C}^\star}\cdot\Gamma$.

We first consider the distribution of Γ, conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$. Since the columns of Γ are statistically independent, we may look at them individually. For each $i\in[n]$, we let $\boldsymbol{\rho}_i^*\in\mathbb{Z}^e$ be an arbitrary solution of $L_{\mathscr{C}^\star}\cdot\boldsymbol{\rho}_i^* = L_{\mathscr{C}^\star}\cdot\boldsymbol{\rho}_i \in \mathbb{Z}_q^{e-1}$. The distribution of $\boldsymbol{\rho}_i\in\mathbb{Z}^e$ conditionally on $\big(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\boldsymbol{\rho}_i\big)$ is $\boldsymbol{\rho}_i^* + D_{\Lambda,\sigma,-\boldsymbol{\rho}_i^*}$, where $\Lambda = \big\{\boldsymbol{x}\in\mathbb{Z}^e \mid L_{\mathscr{C}^\star}\cdot\boldsymbol{x} = \mathbf{0}\big\}$ is the 1-dimensional lattice $\Lambda = \boldsymbol{\kappa}\cdot\mathbb{Z}$.

At this stage, we know each row $\boldsymbol{\rho}_i = (s_i,\rho_{i,2},\ldots,\rho_{i,e})^\top$ of $\Gamma^\top$, conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$, is Gaussian over an affine line. We use this observation to show

that conditioning on $(\mathbf{C}, \mathbf{C}\cdot\Gamma^\top, L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$ is the same as conditioning on $(\mathbf{C}, \mathbf{C}\cdot\mathbf{s}, L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$. In fact, we claim that, conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$, the last $e-1$ columns of $\Gamma^\top$ do not reveal any more information than its first column. Indeed, conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$, each $\boldsymbol{\rho}_i$ can be written $\boldsymbol{\rho}_i = \xi_i\cdot\boldsymbol{\kappa} + \boldsymbol{\rho}_i^*$ for some integer $\xi_i \in \mathbb{Z}$. We may assume that the shifting vector $\boldsymbol{\rho}_i^* = (\rho_{i,1}^*, \ldots, \rho_{i,e}^*)^\top \in \mathbb{Z}_q^e$ is known to $\mathscr{A}$ as it can be obtained from $L_{\mathscr{C}^\star}\cdot\boldsymbol{\rho}_i$ via de-randomized Gaussian elimination. Writing $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_e)$, the j-th column $(\Gamma^\top)_j$ of $\Gamma^\top$ is

$$(\Gamma^\top)_j = \kappa_j \cdot \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_n \end{pmatrix} + \begin{pmatrix} \rho_{1,j}^* \\ \vdots \\ \rho_{n,j}^* \end{pmatrix} \qquad \forall\, j \in [e].$$

As $\kappa_1 = 1$, we have

$$(\Gamma^\top)_j = \kappa_j\cdot(\Gamma^\top)_1 - \kappa_j \cdot \begin{pmatrix} \rho_{1,1}^* \\ \vdots \\ \rho_{n,1}^* \end{pmatrix} + \begin{pmatrix} \rho_{1,j}^* \\ \vdots \\ \rho_{n,j}^* \end{pmatrix} \qquad \forall\, j \in [e].$$

In the latter, the last two terms are information-theoretically known to $\mathscr{A}$ (once we have conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$) and so is κ_j.

We now study the distribution of $\mathbf{s} = (\Gamma^\top)_1$ conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$. By statistical independence, we may consider each coordinate $s_i = (1,0,\ldots,0)^\top\cdot\boldsymbol{\rho}_i$ of $\mathbf{s}$ individually. Recall that, conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$, each $\boldsymbol{\rho}_i$ is distributed as $\boldsymbol{\rho}_i^* + D_{\boldsymbol{\kappa}\mathbb{Z},\sigma,-\boldsymbol{\rho}_i^*}$. Write $\boldsymbol{\rho}_i^* = y\cdot\boldsymbol{\kappa} + (\boldsymbol{\rho}_i^*)^\perp$, with $y\in\mathbb{R}$ and $(\boldsymbol{\rho}_i^*)^\perp$ orthogonal to $\boldsymbol{\kappa}$. Then,

$$\begin{aligned} \boldsymbol{\rho}_i^* + D_{\boldsymbol{\kappa}\mathbb{Z},\sigma,-\boldsymbol{\rho}_i^*} &= \boldsymbol{\rho}_i^* + D_{\boldsymbol{\kappa}\mathbb{Z},\sigma,-y\cdot\boldsymbol{\kappa}-(\boldsymbol{\rho}_i^*)^\perp} \\ &= \boldsymbol{\rho}_i^* + \boldsymbol{\kappa}\cdot D_{\mathbb{Z},\sigma/\|\boldsymbol{\kappa}\|,-y} \end{aligned}$$

We now take the inner product with $(1,0,\ldots,0)$ and use the fact that $\kappa_1 = 1$ to obtain that, conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$, the coordinate s_i is distributed as $(\boldsymbol{\rho}_i^*)_1^\perp + y + D_{\mathbb{Z},\sigma/\|\boldsymbol{\kappa}\|,-y}$. As $\boldsymbol{\kappa}\in\{-1,0,1\}^e$ with the Benaloh-Leichter-based LISS scheme of [DT06], and by our choice of σ, we have that $\sigma/\|\boldsymbol{\kappa}\| = \Omega(\sqrt{n})$. Using Lemma 2.6, this implies that each s_i has min-entropy $\geq \log(\sigma/\|\boldsymbol{\kappa}\|) - 2^{-n} \geq \log\sigma - \frac{1}{2}\log e - 2^{-n}$. Overall, we obtain

$$H_\infty\big(\mathbf{s} \mid L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma\big) \geq n\cdot\log\sigma - \frac{n}{2}\cdot\log e - \frac{n}{2^n}.$$

We are now ready to conclude. By the above, to prove the claim (and hence the lemma), it suffices to obtain a lower bound on the min-entropy of $\mathbf{s}$ conditioned on $(\mathbf{C}, \mathbf{C}\cdot\mathbf{s}, L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$. We then use the above min-entropy lower bound on $\mathbf{s}$ conditioned on $(L_{\mathscr{C}^\star}, L_{\mathscr{C}^\star}\cdot\Gamma)$ and the fact that given $\mathbf{C}$, the quantity $\mathbf{C}\cdot\mathbf{s}\in\mathbb{Z}_q^{n'}$ reveals at most $n'\log q$ bits. □

3.6 Robustness from Homomorphic Signatures

In this section, we show how to extend our DPRF to achieve robustness against faulty servers. To this end, we follow the approach of [BGGK17, BGG+18] which relies on homomorphic signatures to obtain a mechanism allowing to verify that servers' computations are carried out correctly.

In short, a leveled homomorphic signature makes it possible to publicly derive a signature Φ_C on a circuit evaluation $C(M)$, given a signature Φ on an initial message M. In our setting, as well as in [BGGK17, BGG+18], we need a homomorphic signature which is *context-hiding*, meaning that a homomorphically evaluated signature Φ_C is statistically independent of the initial message M: more formally, there should be a simulator that creates Φ_C from scratch from the signing key sk and the value $C(M)$ and, yet, the joint distribution of $(C(M), \Phi_C)$ should be statistically close to that obtained by homomorphically deriving Φ_C from (M, Φ). One difficulty is that the fully homomorphic signatures of [GVW15b] are not known to be simultaneously context-hiding and adaptively unforgeable (i.e., context-hiding security was only achieved for selectively unforgeable schemes). Fortunately, selective unforgeability is sufficient for our purposes (and those of [BGGK17]). We can thus instantiate our construction using the context-hiding construction of Gorbunov, Vaikuntanathan and Wichs [GVW15b], which relies on the Short-Integer-Solution (SIS) assumption. Since the LWE assumption implies the SIS assumption, we can achieve robustness without introducing any additional assumption, analogously to [BGGK17, BGG+18].

A Robust DPRF is a DPRF endowed with an additional verification algorithm Rob.Verify that takes as input the public parameters of the DPRF along with an input X and a candidate partial evaluation Y_j on behalf of server $j \in [N]$, together with a corresponding label ϕ_j. It outputs 1 or 0 depending on whether (Y_j, ϕ_j) is deemed valid or not. For correctness, we require that $\mathsf{Rob.Verify}(\mathsf{pp}, Y_j, \phi_j, X) = 1$ for all $j \in [N]$, $\mathsf{pp} \leftarrow \mathsf{Rob.Setup}(1^\lambda, 1^\ell, 1^\mathsf{t}, 1^\mathsf{N})$, any input $X \in \mathscr{X}$ and any evaluation $(Y_j, \phi_j) \leftarrow \mathsf{Rob.PEval}(\mathsf{pp}, SK_j, X)$, for $(SK_1, \ldots, SK_N) \leftarrow \mathsf{Rob.Share}(SK_0)$ obtained by sharing $SK_0 \leftarrow \mathscr{K}$. The robustness property requires that it be unfeasible for a corrupted server $j \in [N]$ holding the secret share SK_j to produce an incorrect partial evaluation Y_j, for some input X, that still passes the verification $\mathsf{Rob.Verify}(\mathsf{pp}, Y_j, X) = 1$.

Definition 3.6 (Robust DPRF)**.** *A robust DPRF family is specified by the algorithms:* $(\mathsf{Rob.Setup}, \mathsf{Rob.Keygen}, \mathsf{Rob.Share}, \mathsf{Rob.PEval}, \mathsf{Rob.Eval}, \mathsf{Rob.Combine})$, *together with a polynomial-time verification algorithm,* Rob.Verify*, such that:*

Correctness: *For any* $\mathsf{pp} \leftarrow \mathsf{Rob.Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$, $SK_0 \leftarrow \mathsf{Rob.Keygen}(\mathsf{pp})$, *any sharing* $(SK_1, \ldots, SK_N) \leftarrow \mathsf{Rob.Share}(\mathsf{SK}_0)$, *any index* $j \in [N]$, *any* $X \in \mathscr{X}$ *and any* $(Y_j, \phi_j) \leftarrow \mathsf{Rob.PEval}(\mathsf{pp}, SK_j, X)$, *we have* $\mathsf{Rob.Verify}(\mathsf{pp}, j, Y_j, \phi_j, X) = 1$.

Robustness: *For any $j \in [N]$ and any* PPT *adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that:*

$$\Pr\left[\mathsf{Rob.Verify}(\mathsf{pp}, j, \tilde{Y}_j, \tilde{\phi}_j, \tilde{X}) = 1 \;\middle|\; \begin{array}{c}(\tilde{Y}_j, \tilde{\phi}_j, \tilde{X}) \leftarrow \mathscr{A}(\mathsf{pp}, \boldsymbol{SK}) \\ \wedge \\ \tilde{Y}_j \neq Y_j\end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over: the randomness of $\mathscr{A}$, and over the randomness: $(Y_j, \phi_j) \leftarrow \mathsf{Rob.PEval}(SK_j, \tilde{X})$, $\mathsf{pp} \leftarrow \mathsf{Rob.Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$ *and* $SK_0 \leftarrow \mathsf{Rob.Keygen}(\mathsf{pp})$ *and* $\boldsymbol{SK} := (SK_1, \ldots, SK_N) \leftarrow \mathsf{Rob.Share}(SK_0)$.

3.6.1 Homomorphic Signatures

In this section we recall the definition of leveled homomorphic signatures (HS) from [GVW15b]. We use the simplified version presented in [BGGK17].

Definition 3.7 (HS)**.** *A (leveled) homomorphic signature scheme is a tuple of efficient algorithms* $\mathsf{HS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignEval}, \mathsf{Verify})$ *with the following specifications.*

KeyGen$(1^\lambda, 1^d, 1^K)$**:** *On input the security parameter λ, a circuit depth bound d and a data set bound K, the algorithm outputs a signing key and a verification key, (sk, vk) .*

Sign$(sk, \boldsymbol{m})$**:** *On input of the signing key sk and a message $\boldsymbol{m} \in \{0,1\}^K$, the algorithm outputs a signature Φ.*

SignEval(C, Φ)**:** *On input of a circuit $C : \{0,1\}^K \to \{0,1\}$ and a signature, the algorithm outputs a homomorphically evaluated signature $\Phi^\star$.*

Verify$(vk, C, y, \Phi^\star)$**:** *Given a verification key vk, a circuit $C : \{0,1\}^K \to \{0,1\}$, an output value y and a signature $\Phi^\star$, the algorithm outputs* 1 *or* 0.

Correctness: For all $\lambda, d, K \in \mathbb{N}$, $(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\mathsf{d}, 1^\mathsf{K})$, $\boldsymbol{m} \in \{0,1\}^K$, $\Phi \leftarrow \mathsf{Sign}(sk, \boldsymbol{m})$, $C : \{0,1\}^K \to \{0,1\}$ a circuit of depth at most d and $y = C(\boldsymbol{m})$, if $\Phi^\star \leftarrow \mathsf{SignEval}(C, \Phi)$, then $\mathsf{Verify}(vk, C, y, \Phi^\star) = 1$.

For our purposes, we need a homomorphic signature that satisfies two security requirements: *unforgeability* and *context-hiding* [GVW15b], that we present below.

Definition 3.8 (Selective Unforgeability)**.** *We say that a leveled homomorphic signature scheme* $\mathsf{HS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignEval}, \mathsf{Verify})$ *provides selective unforgeability if, for any* PPT *adversary $\mathscr{A}$, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathbf{Adv}^{\textit{uf-HS}}_{\mathscr{A}}(\lambda) := \Pr[\mathsf{Exp}^{\textit{uf-HS}}_{\mathscr{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}^{\textit{uf-HS}}_{\mathscr{A}}(\lambda)$ *is defined below:*

1. $(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^\mathsf{d}, 1^\mathsf{K})$
2. $\boldsymbol{m}^\star \leftarrow \mathscr{A}(1^\lambda, 1^d, 1^K)$
3. $\Phi \leftarrow \mathsf{Sign}(sk, \boldsymbol{m}^\star)$
4. $(C^\star, y^\star, \Phi^\star) \leftarrow \mathscr{A}(\Phi, vk)$

The experiment outputs 1 *if and only if the following conditions are satisfied:*

- *The depth of* $C^\star$ *is at most* d
- $C^\star(\boldsymbol{m}^\star) \neq y^\star$
- $\mathsf{Verify}(vk, C^\star, y^\star, \Phi^\star) = 1$.

Definition 3.9 (Context-hiding)**.** *A leveled homomorphic signature scheme* $\mathsf{HS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{SignEval}, \mathsf{Verify})$ *is context-hiding if there exists a simulator* Sim *such that, for any key* $(sk, vk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^d, 1^K)$*, any* $\boldsymbol{m} \in \{0,1\}^K$*, any signature* $\Phi \leftarrow \mathsf{Sign}(sk, \boldsymbol{m})$*, and any circuit* C *of depth* $\leq d$*, the following distributions are statistically close:*

$$\mathsf{SignEval}(C, \Phi) \overset{s}{\approx} \mathsf{Sim}(sk, C, C(\boldsymbol{m}))$$

where the randomness is taken over the random coins of Sim *and those of* $\mathsf{SignEval}$.

3.6.2 A Robust DPRF Construction

In this section, we use homomorphic signatures to make our DPRF construction of Section 3.5 robust against malicious servers. For this purpose, we need a leveled homomorphic signatures [GVW15b] that is context-hiding. Following the approach of [BGGK17, BGG+18], we have the trusted dealer provide each server $j \in [N]$ with a secret key share SK_j together with a homomorphic signature Φ_j of that secret share. By leveraging the homomorphism of the signature scheme, each server j can derive (without knowing the signing key) valid signatures for circuits C_X that evaluate $Y_j = \mathsf{PEval}(SK_j, X)$ on input of SK_j. Each server can thus produce a valid signature for the partial evaluation of the PRF on any input X using its own share SK_j and the corresponding signature. The unforgeability of homomorphic signatures makes it unfeasible to come up with a valid signature on an incorrect partial evaluation of the PRF for the input X.

Let a DPRF specified by algorithms $(\mathsf{DPRF.Setup}, \mathsf{DPRF.Share}, \mathsf{DPRF.PEval}, \mathsf{DPRF.Eval}, \mathsf{DPRF.Combine})$, as described in Section 3.5. Let a homomorphic signature scheme $(\mathsf{HS.KeyGen}, \mathsf{HS.Sign}, \mathsf{HS.SignEval}, \mathsf{HS.Verify})$. We now describe a robust DPRF as follows.

Rob.Setup$(1^\lambda, 1^\ell, 1^t, 1^N)$**:** Run $\mathsf{pp} \leftarrow \mathsf{DPRF.Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$. Define the input space $\mathscr{X}$ and the key space $\mathscr{K}$ to be the same as in the underlying DPRF. Let K the number of bits to represent any element of $\mathscr{K}$ and d is the maximal depth of any Boolean circuit that computes $C_X(\cdot) := \mathsf{DPRF.PEval}(\cdot, X)$ for any input $X \in \mathscr{X}$. Also run $(sk_j, vk_j) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^d, 1^K)$ for each $j \in [N]$.

Return $\mathsf{Rob.pp} = \big(\mathsf{pp}, \{vk_j\}_{j\in[N]}, 1^d\big)$.

Rob.Keygen(pp)**:** Run $SK_0 \leftarrow \mathsf{DPRF.Keygen}(\mathsf{pp})$, output SK_0.

Rob.Share(SK_0)**:** In order to share $SK_0 \in \mathscr{K}$, and do the following.

1. Run $(SK_1, \ldots, SK_N) \leftarrow \mathsf{DPRF.Share}(SK_0)$.
2. Compute $\Phi_j \leftarrow \mathsf{HS.Sign}(sk_j, SK_j)$ for all $j \in [N]$.
   Output $\big((SK_1, \Phi_1), \ldots, (SK_N, \Phi_N)\big)$

For each $j \in [N]$, server j receives the pair (SK_j, Φ_j). The signing keys sk_j are the ones generated by the Rob.Setup algorithm.

Rob.PEval$((SK_j, \Phi_j), X)$**:** Compute $Y_j \leftarrow \mathsf{DPRF.PEval}(SK_j, X)$. Then, compute $\Phi_{j,X} \leftarrow \mathsf{HS.SignEval}(C_X, \Phi_j)$, where C_X is a Boolean circuit such that $C_X(\cdot) = \mathsf{DPRF.PEval}(\cdot, X)$; Then, return $(Y_j, \Phi_{j,X})$.

Rob.Eval(SK_0, X)**:** Compute $Y \leftarrow \mathsf{DPRF.Eval}(SK_0, X)$.

Rob.Combine$(\mathscr{S}, (Y_{j_1}, \Phi_{j_1}), \ldots, (Y_{j_t}, \Phi_{j_t}))$**:** On input a t-subset of $[N]$ and t partial evaluations, output $Y \leftarrow \mathsf{DPRF.Combine}(\mathscr{S}, (Y_{j_1}, \ldots, Y_{j_t}))$.

Rob.Verify$(\mathsf{Rob.pp}, j, (Y_j, \Phi_{j,X}), X)$**:** If $\mathsf{HS.Verify}\big(vk_j, C_X, Y_j, \Phi_{j,X}\big) = 1$, return 1. If not, return 0.

The correctness of the Rob.Verify algorithm follows from the correctness of the homomorphic signature scheme. The robustness of the DPRF is implied by the selective unforgeability of the homomorphic signature. From a robustness adversary, we can easily construct a selective forger in the sense of Definition 3.8 as the only way for the adversary to trick the verification algorithm into accepting an incorrect partial evaluation is to break the unforgeability of the homomorphic signature. Moreover, the latter only needs to provide selective unforgeability since, in the reduction, the adversary only needs to see one verification key vk after the generation of signatures on secret key shares $\{SK_j\}_{j\in[N]}$.

Theorem 3.8. *Assuming that the homomorphic signature scheme provides selective unforgeability, the above construction satisfies the robustness property.*

Proof. Towards a contradiction, suppose that there exists a polynomial-time adversary $\mathscr{A}$ that breaks the robustness for the index $j \in [N]$(in the sense of Definition 3.6) of the Rob.DPRF. We build a polynomial-time adversary $\mathscr{B}$ that can forge signatures, thus breaking the selective unforgeability of the homomorphic signature HS.

The reduction $\mathscr{B}$ starts by running $\mathsf{pp} \leftarrow \mathsf{DPRF.Setup}(1^\lambda, 1^\ell, 1^t, 1^N)$ such that any key in $\mathscr{K}$ can be represented using K bits and, for any input $X \in \mathscr{X}$, the Boolean circuit $C_X(\cdot) := \mathsf{DPRF.PEval}(\cdot, X)$ has depth at most d. Next, the reduction $\mathscr{B}$ generates a master secret key $SK_0 \leftarrow \mathsf{DPRF.Keygen}(\mathsf{pp})$ and computes $\boldsymbol{SK} = (SK_1, \ldots, SK_N)$ by running the sharing algorithm $\mathsf{DPRF.Share}(SK_0)$.

To finalize step 2 of the selective unforgeability game, $\mathscr{B}$ sends $\boldsymbol{m}^\star := SK_j \in \{0,1\}^K$ to the challenger as the signing query. The challenger runs $(vk_j, sk_j) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^d, 1^K)$ and provides $\mathscr{B}$ with the public verification key vk_j together with $\Phi_j \leftarrow \mathsf{HS.Sign}(sk_j, \boldsymbol{m}^\star)$. Moreover, $\mathscr{B}$ runs $(sk_i, vk_i) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^d, 1^K)$ for all $i \in [N] \setminus \{j\}$ and computes the rest of the signatures $\Phi_i \leftarrow \mathsf{HE.Sign}(sk_i, SK_i)$, for all $i \in [N] \setminus \{j\}$. Now, $\mathscr{B}$ is able to compute both the public parameters of the Rob.DPRF as $(\mathsf{pp}, \{vk_j\}_{j\in[N]}$ and the vector $\boldsymbol{SK} := ((SK_1, \Phi_1), \ldots, (SK_N, \Phi_N))$, which corresponds to all the secret shares of the Rob.DPRF scheme. Then it gives both to the attacker $\mathscr{A}$.

Since $\mathscr{A}$ can break robustness, by hypothesis, it has non-negligible probability of outputting $(\tilde{Y}_j, \tilde{\Phi}_j^\star, \tilde{X})$ such that $\mathsf{Rob.Verify}(\mathsf{pp}, j, (\tilde{Y}_j, \tilde{\Phi}_j^\star), \tilde{X}) = 1$ and $\tilde{Y}_j$ is not obtained by running $\mathsf{Rob.PEval}((SK_j, \phi_j), \tilde{X})$. By construction, this translates for the homomorphic signature scheme into:

$$\mathsf{HS.Verify}(vk_j, C_{\tilde{X}}, \tilde{Y}_j, \tilde{\Phi}_j^\star) = 1 \qquad \wedge \qquad \tilde{Y}_j \neq \mathsf{DPRF.PEval}(SK_j, \tilde{X}),$$

where the second condition is equivalent to $\tilde{Y}_j \neq C_{\tilde{X}}(SK_j)$.

At step 4 of the selective unforgeability game, algorithm $\mathscr{B}$ outputs a triple $(C_{\tilde{X}}, \tilde{Y}_j, \tilde{\Phi}_j^\star)$, which constitutes a forgery for the signature scheme since the verification algorithm accepts even though $C_{\tilde{X}}(\boldsymbol{m}^\star) \neq \tilde{Y}_j$. The probability that $\mathscr{B}$ outputs a forgery is the same as the probability of $\mathscr{A}$ in breaking the robustness of the Rob.DPRF. □

We still have to prove that above construction preserves the underlying DPRF's security under adaptive corruptions.

Theorem 3.9. *If the underlying DPRF is adaptively secure and the homomorphic signature scheme is context-hiding, then the construction above is an adaptively secure robust DPRF.*

Proof. Suppose that there exists a polynomial-time adversary $\mathscr{A}$ that wins the adaptive security game for the above construction. We build an efficient adversary $\mathscr{B}$ that wins the adaptive security game for the underlying (non-robust) DPRF.

Algorithm $\mathscr{B}$ runs $\mathscr{A}$ on input of the public parameters pp that it receives from its own challenger. The reduction $\mathscr{B}$ also generates N homomorphic signature

key pairs $(sk_i, vk_i) \leftarrow \mathsf{HS.KeyGen}(1^\lambda, 1^d, 1^K)$, for $i \in [N]$, where d is the maximum depth of the Boolean circuits $C_X(\cdot) = \mathsf{PEval}(\cdot, X)$, for any input $X \in \mathscr{X}$. (The input space and the key space are known from the public parameters pp). The public parameters for the robust construction, $\big(\mathsf{pp}, \{vk_i\}_{i\in[N]}\big)$, are given to $\mathscr{A}$.

For each corruption query $j \in [N]$ made by $\mathscr{A}$, the reduction $\mathscr{B}$ relays the query to its DPRF challenger. Upon receiving SK_j from the latter, $\mathscr{B}$ computes $\Phi_j := \mathsf{HS.Sign}(sk_j, SK_j)$ and hands (SK_j, Φ_j) to $\mathscr{A}$. To answer a partial evaluation query $(i, X) \in [N] \times \{0,1\}^\ell$ made by $\mathscr{A}$, the reduction $\mathscr{B}$ sends the same query to its challenger and, upon receiving the response Y_i, it computes $\Phi_i^\star = \mathsf{Sim}(sk_i, C_X, Y_i)$ and returns the pair $(Y_i, \Phi_i^\star)$ to $\mathscr{A}$. The context hiding property of the homomorphic signature scheme ensures that simulated signatures $\Phi_i^\star$ are statistically indistinguishable from an honestly derived signature of the form $\mathsf{HS.SignEval}(C_X, \Phi_i)$, where $\Phi_i = \mathsf{HS.Sign}(sk, SK_i)$ (note that $\mathscr{B}$ has to run the context-hiding simulator here since it does not know i's share (SK_i, Φ_i)).

For the challenge query $X^\star$ made by $\mathscr{A}$, $\mathscr{B}$ sends the same query to its own DPRF challenger and forwards the latter's challenge $Y^\star$ to $\mathscr{A}$. At the end of the game, $\mathscr{B}$ outputs whatever $\mathscr{A}$ outputs. It is easy to see that, by construction, $\mathscr{B}$ succeeds whenever $\mathscr{A}$ does. □

Chapter

4

Multi-Client Functional Encryption from LWE

The focus of this chapter is the private-key Multi-Client Functional Encryption scheme of [LT19] for the bounded inner-product class of functionalities. In the multi-input setting, this was the first construction with labels, that was proved secure against adaptive corruptions, in the standard model. The security relies on the LWE assumption with super-polynomial modulus q, due to the use of the rounding techniques introduced in [BPR12].

Our observation is that we can construct MCFE for the class of linear functions from Key-Homomorphic PRFs. Directly leveraging the pseudo-randomness of the KH-PRF only yields secure MCFEs in the selective security model, in which the adversary has to announce its challenge in advance.

Our idea is to adapt the KH-PRF construction of [BLMR13] and use its particular design to construct adaptively secure MCFE, in a non-modular way. We employed the same high-level strategy to get adaptive security, in a non-modular way, in Chapter 3.

Organization

We start section 4.1 by giving the formal definitions for the MCFE primitive and comparing different security notions. In section 4.2.1 we highlight the connection between MCFEs and KH-PRFs and discuss the selectively secure generic MCFE construction that arises from it. The main result of this chapter is presented in section 4.3, where we give an adaptively secure MCFE based on the LWE assumption. Finally, we can upgrade the security of our scheme by using the compiler presented in Section 4.4.

4.1 Definitions

First we recall the syntax of multi-client functional encryption as introduced in [GKL+14].

4.1.1 Multi-Client Functional Encryption (MCFE)

Definition 4.1. *A multi-client functional encryption (MCFE) scheme for a message space $\mathcal{M}$ and tag space $\mathcal{T}$ is a tuple* $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{DKeygen}, \mathsf{Decrypt})$ *of efficient algorithm with the following specifications:*

Setup$(\mathsf{cp}, 1^\ell)$: *Takes in global parameters* cp *and a pre-determined number of users* 1^ℓ*, where* cp *specifies a security parameter* 1^λ*. It outputs a set of public parameters* mpk*, a master secret key* msk*, and a set of encryption keys* $\{\mathsf{ek}_i\}_{i=1}^{\ell}$*. We assume that* mpk *is included in all encryption keys* ek_i*.*

Encrypt(ek_i, x_i, t) : *Takes as input the encryption key* ek_i *of user* $i \in [\ell]$*, a message* x_i *and a tag* $t \in \mathcal{T}$*. It output a ciphertext* $C_{t,i}$*.*

DKeygen(msk, f) : *Takes as input the master secret key* msk *and an* ℓ*-argument function* $f : \mathcal{M}^\ell \to \mathcal{R}$*. It outputs a functional decryption key* dk_f*.*

Decrypt$(\mathsf{dk}_f, t, \mathbf{C})$: *Takes as input a functional decryption key* dk_f*, a tag* t*, and an* ℓ*-vector of ciphertexts* $\mathbf{C} = (C_{t,1}, \ldots, C_{t,\ell})$*. It outputs a function evaluation* $f(\boldsymbol{x}) \in \mathcal{R}$ *or an error message* $\perp$*.*

Correctness. For any public parameters cp, $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\ell)$, any vector $\boldsymbol{x} \in \mathcal{M}^n$ any tag $t \in \mathcal{T}$ and any function $f : \mathcal{M}^\ell \to \mathcal{R}$, for all $i \in [\ell]$ $C_{t,i} \leftarrow \mathsf{Encrypt}(\mathsf{ek}_i, x_i, t)$ and $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$, we have $\mathsf{Decrypt}\big(\mathsf{dk}_f, t, \mathbf{C}_t = (C_{t,1}, \ldots, C_{t,\ell})\big) = f(\boldsymbol{x})$ with overwhelming probability.

4.1.2 Security

We now recall the security definition given in [CDG+18a], for an adaptively secure MCFE. Then we give another equivalent definition for the same security notion,

that we use in the main proof. Both definitions are given as indistinguishability (IND) games between a challenger and an adversary.

Definition 4.2 (IND-security)**.** *For an MCFE scheme with ℓ senders, consider the following game between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The game involves a set $\mathscr{HS}$ of honest senders (initialized to $\mathscr{HS} := [\ell]$) and a set $\mathscr{CS}$ (initialized to $\mathscr{CS} := \emptyset$) of corrupted senders.*

Initialization: *The challenger $\mathscr{C}$ runs* $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^{\ell})$. *Then, it chooses a random bit $b \leftarrow \{0,1\}$ and gives the master public key,* mpk, *to the adversary*

Encryption queries: *The adversary $\mathscr{A}$ can adaptively make encryption queries of the form* $\mathsf{QEncrypt}(i, x^0, x^1, t)$, *to which the challenger replies with the encryption* $\mathsf{Encrypt}(\mathsf{ek}_i, x^b, t)$. *For any given pair (i,t), only one query is allowed and subsequent queries involving the same (i,t) are ignored.*

Functional decryption key queries: *The adversary can adaptively obtain decryption keys by making queries of the form* $\mathsf{QDKeygen}(f)$. *The challenger returns* $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$.

Corruption queries: *For any user $i \in \mathscr{HS}$, the adversary can adaptively make queries* $\mathsf{QCorrupt}(i)$, *to which the challenger replies with* ek_i *and updates $\mathscr{HS}$ and $\mathscr{CS}$ by setting $\mathscr{CS} := \mathscr{CS} \cup \{i\}$ and $\mathscr{HS} := \mathscr{HS} \setminus \{i\}$.*

Finalize: *The adversary makes its guess $b' \in \{0,1\}$; $\mathscr{A}$ wins the game if $\beta = b$, where β is defined to be $\beta := b'$ except in the following situations.*

1. *An encryption query* $\mathsf{QEncrypt}(i, x^0, x^1, t)$ *has been made for an index $i \in \mathscr{CS}$ with $x^0 \neq x^1$.*
2. *For some label t, an encryption query* $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *has been asked for $i \in \mathscr{HS}$, but encryption queries* $\mathsf{QEncrypt}(j, x_j^0, x_j^1, t)$ *have not been asked for all $j \in \mathscr{HS}$.*
3. *For a label t and some function f queried to* $\mathsf{QDKeygen}$, *there exists a pair of vectors $(\boldsymbol{x}^0, \boldsymbol{x}^1)$ such that $f(\boldsymbol{x}^0) \neq f(\boldsymbol{x}^1)$, where*
   - *$x_i^0 = x_i^1$ for all $i \in \mathscr{CS}$;*
   - $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ *have been asked for all $i \in \mathscr{HS}$.*

In any of the above cases, $\mathscr{A}$'s output is replaced by a random $\beta \leftarrow U(\{0,1\})$.

An MCFE scheme provides IND *security if, for any* PPT *adversary $\mathscr{A}$, we have* $\mathbf{Adv}_{\mathscr{A}}^{MCFE}(\lambda) := \left|\Pr[\beta = 1 \mid b = 1] - \Pr[\beta = 1 \mid b = 0]\right| \in \mathsf{negl}(\lambda)$.

In this chapter, it will be convenient to work with the following security definition, which is equivalent to Definition 4.2, and the equivalence is proved below in Proposition 4.2.

Definition 4.3 (1-challenge-IND-security)**.** *For an MCFE scheme with ℓ senders, we consider the following game between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The game involves a set $\mathscr{HS}$ (initialized to $\mathscr{HS} := [\ell]$), of honest senders and a set $\mathscr{CS}$ (initialized to $\mathscr{CS} := \emptyset$), of corrupted senders.*

Initialization: *The challenger $\mathscr{C}$ runs* $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^{\ell})$ *for the chosen global parameters* cp*. Then, it chooses a random bit $b \leftarrow \{0,1\}$ and gives the master public key* mpk *to the adversary $\mathscr{A}$.*

Encryption queries: *The adversary can adaptively make encryption queries of the form* $\mathsf{QEncrypt}(i, x, t)$*, to which the challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x, t)$*. Any further query involving the same pair (i, t) is ignored.*

Challenge queries: *The adversary adaptively makes challenge queries of the form* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^{\star})$*. The challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^{\star b}, t^{\star})$*. Only one tag $t^{\star}$ can be involved in a challenge query. If $t^{\star}$ denotes the tag of the first query, the challenger only replies to subsequent challenge queries for the same label $t^{\star}$. Moreover, only one query $(i, t^{\star})$ is allowed for each $i \in [\ell]$ and subsequent queries involving the same $i \in [\ell]$ are ignored.*

Functional decryption key queries: *The adversary can adaptively obtain decryption keys via queries* $\mathsf{QDKeygen}(f)$*. At each query, the challenger returns* $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$*.*

Corruption queries: *For any user $i \in \mathscr{HS}$, the adversary can adaptively make queries* $\mathsf{QCorrupt}(i)$*, to which the challenger replies with* ek_i *and updates $\mathscr{HS}$ and $\mathscr{CS}$ by setting $\mathscr{CS} := \mathscr{CS} \cup \{i\}$ and $\mathscr{HS} := \mathscr{HS} \setminus \{i\}$.*

Finalize: *The adversary outputs a bit $b' \in \{0,1\}$. The adversary $\mathscr{A}$ wins if $\beta = b$, where β is defined as $\beta := b'$, unless of the situations below occurred.*

1. *A challenge query* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^{\star})$ *has been made for an index $i \in \mathscr{CS}$ with $x_i^{\star 0} \neq x_i^{\star 1}$.*
2. *An encryption query* $\mathsf{QEncrypt}(i, x, t^{\star})$ *has been made for the challenge tag $t^{\star}$ for some index $i \in [\ell]$.*
3. *For the challenge tag $t^{\star}$, a challenge query* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^{\star})$ *has been asked for some index $i \in \mathscr{HS}$, but not all the challenge queries* $\mathsf{CQEncrypt}(j, x_j^{\star 0}, x_j^{\star 1}, t^{\star})$ *have not been asked for all $j \in \mathscr{HS}$.*
4. *For the challenge tag $t^{\star}$ and some function f queried to* QDKeygen*, there exists a pair of vectors $(\boldsymbol{x}^{\star 0}, \boldsymbol{x}^{\star 1})$ such that $f(\boldsymbol{x}^{\star 0}) \neq f(\boldsymbol{x}^{\star 1})$, where*
   - *$x_i^{\star 0} = x_i^{\star 1}$ for all $i \in \mathscr{CS}$;*
   - *$\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^{\star})$ have been asked for all $i \in \mathscr{HS}$.*

If any of these events occurred, $\mathscr{A}$'s output is overwritten by $\beta \leftarrow U(\{0,1\})$.

We say that an MCFE scheme is 1ch-IND secure if, for any PPT *adversary* $\mathscr{A}$*, we have* $\mathbf{Adv}^{1ch\text{-}IND}_{\mathscr{A}}(\lambda) := \left|\Pr[\beta = b] - \frac{1}{2}\right| \in \mathsf{negl}(\lambda)$.

Condition 1 and condition 4 from the above **Finalize** step are necessary to prevent the adversary to trivially win the indistinguishability game. Also the adversary is only allowed to make one encryption or challenge query per $(\mathsf{slot}, \mathsf{label})$-pair (specified in the syntax and by condition 2). Without condition 2 the adversary could easily win the game for any scheme that only supports deterministic encryption. We can actually relax condition 2 by the following, which allows both encryption and challenge queries on the pair $(i, t^\star)$ as long as $x_i^{0\star} = x_i^{1\star}$, while maintaining the same level of security:

2′. Both $\mathsf{QEncrypt}(i, x, t^\star)$ and $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ have been made for an index i and the challenge label $t^\star$, such that $x_i^{0\star} \neq x_i^{1\star}$

Proposition 4.1 below shows that replacing condition 2 by condition 2′ does not make the adversary any stronger. We say that a scheme is 1ch-IND′ secure if it satisfies the above security definition with condition 2 replaced by condition 2′.

The most artificial condition in the **Finalize** step is condition 3, which stops the adversary from using any information on partial ciphertexts. We require this condition since we need it in the security proof (Theorem 4.2). In section 4.4 we give a compiler that works in the standard model and upgrades the security of any 1-challenge-IND secure MCFE, by removing the unnatural condition 3 from the security definition.

Next, we give the formal definition of an MCFE scheme without the artificial condition 3. We call this notion *1-or-less-IND* security, because the adversary can choose not to make or to make at most one challenge query for every honest slot.

Definition 4.4 (1-or-less-IND-security)**.** *For an MCFE scheme with ℓ senders, we consider the following game between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The game involves a set $\mathscr{HS}$ (initialized to $\mathscr{HS} := [\ell]$), of honest senders and a set $\mathscr{CS}$ (initialized to $\mathscr{CS} := \emptyset$), of corrupted senders.*

Initialization: *The challenger $\mathscr{C}$ runs* $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\ell)$ *for the chosen global parameters* cp*. Then, it chooses a random bit $b \leftarrow \{0,1\}$ and gives the master public key* mpk *to the adversary $\mathscr{A}$.*

Encryption queries: *The adversary can adaptively make encryption queries of the form* $\mathsf{QEncrypt}(i, x, t)$*, to which the challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x, t)$*. Any further query involving the same pair (i, t) is ignored.*

Challenge queries: *The adversary adaptively makes challenge queries of the form* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$*. The challenger replies with* $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^{\star b}, t^\star)$*. Only one tag $t^\star$ can be involved in a challenge query. If $t^\star$ denotes the tag of the first query, the challenger only replies to subsequent challenge queries for the same label $t^\star$. Moreover, only one query $(i, t^\star)$ is allowed for each $i \in [\ell]$ and subsequent queries involving the same $i \in [\ell]$ are ignored.*

Functional decryption key queries: *The adversary can adaptively obtain decryption keys via queries* $\mathsf{QDKeygen}(f)$. *At each query, the challenger returns* $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f)$.

Corruption queries: *For any user* $i \in \mathcal{HS}$, *the adversary can adaptively make queries* $\mathsf{QCorrupt}(i)$, *to which the challenger replies with* ek_i *and updates* $\mathcal{HS}$ *and* $\mathcal{CS}$ *by setting* $\mathcal{CS} := \mathcal{CS} \cup \{i\}$ *and* $\mathcal{HS} := \mathcal{HS} \setminus \{i\}$.

Finalize: *The adversary outputs a bit* $b' \in \{0,1\}$. *The adversary* $\mathcal{A}$ *wins if* $\beta = b$, *where* β *is defined as* $\beta := b'$, *unless of the situations below occurred.*

1. *A challenge query* $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ *has been made for an index* $i \in \mathcal{CS}$ *with* $x_i^{\star 0} \neq x_i^{\star 1}$.
2. *An encryption query* $\mathsf{QEncrypt}(i, x, t^\star)$ *has been made for the challenge tag* $t^\star$ *for some index* $i \in [\ell]$.
4. *For the challenge tag* $t^\star$ *and some function* f *queried to* $\mathsf{QDKeygen}$, *there exists a pair of vectors* $(\boldsymbol{x}^{\star 0}, \boldsymbol{x}^{\star 1})$ *such that* $f(\boldsymbol{x}^{\star 0}) \neq f(\boldsymbol{x}^{\star 1})$, *where*
   - $x_i^{\star 0} = x_i^{\star 1}$ *for all* $i \in \mathcal{CS}$;
   - $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ *have been asked for all* $i \in \mathcal{HS}$.

If any of these events occurred, $\mathcal{A}$*'s output is overwritten by* $\beta \leftarrow U(\{0,1\})$.

We say that an MCFE scheme provides 1-or-less security if, for any PPT *adversary* $\mathcal{A}$, *we have* $\mathbf{Adv}^{\text{1-or-less}}_{\mathcal{A}}(\lambda) := \left|\Pr[\beta = b] - \frac{1}{2}\right| \in \mathsf{negl}(\lambda)$.

The definition of 1-or-less security is identical to the 1-challenge-IND definition, except that we removed Condition 3 of the Finalize step in Definition 4.3. In Definition 4.4, we insist that the last condition of the Finalize step does not impose any restriction on the functions queried to QDKeygen if there exists a single index $i \in \mathcal{HS}$ for which no challenge query $\mathsf{CQEncrypt}(i, \cdot, \cdot, t^\star)$ was made.

Proposition 4.1. *1ch-IND security is equivalent to 1ch-IND′ security.*

Proof. Recall that 1ch-IND′ refers to the same security notion as described in Definition 4.3, except that Condition 2 in the Finalize step is replaced by

> 2′: Both $\mathsf{QEncrypt}(i, x, t^\star)$ and $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$ have been made for an index i and the challenge label $t^\star$, such that $x_i^{0\star} \neq x_i^{1\star}$.

The non-trivial implication to prove is that 1ch-IND security implies 1ch-IND′ security. Towards a contradiction, suppose that there exists an adversary $\mathcal{A}'$ that wins the 1ch-IND′ game with noticeable advantage. We build an adversary $\mathcal{A}$ that breaks the 1ch-IND security game. Our adversary $\mathcal{A}$ answers both corruption and functional decryption key queries made by $\mathcal{A}'$ by relaying them to its challenger and forwarding the responses to $\mathcal{A}'$. For a query $\mathsf{QEncrypt}(i, x, t)$ made by $\mathcal{A}'$, $\mathcal{A}$

checks if a challenge query $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t)$ was made earlier. If no challenge query has been made so far for the pair (i, t), the reduction sends the same $\mathsf{QEncrypt}(i, x, t)$ query to its challenger and passes the the answer to $\mathscr{A}'$. Otherwise, it makes a $\mathsf{Corrupt}(i)$ query to its challenger and thereby obtains the encryption key ek_i. To answer any subsequent query for slot i, the reduction simply uses ek_i.

To answer a challenge query $\mathsf{QCEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$, the reduction $\mathscr{A}$ forwards the query to its challenger and transmits the answer to $\mathscr{A}$' if it did not previously make an encryption $\mathsf{QEncrypt}(i, x, t^\star)$ for the pair $(i, t^\star)$. If $\mathscr{A}$ previously made an encryption query for the pair $(i, t^\star)$, it also invokes its challenger and sends it the query $\mathsf{Corrupt}(i)$. Upon receiving ek_i, $\mathscr{A}$ is able to answer any query concerning the i-th slot.

Notice that the adversary $\mathscr{A}'$ is only allowed to make both encryption and challenge queries on the same pair $(i, t^\star)$ if $x_i^{\star 0} = x_i^{\star 1}$. This implies that the reduction is allowed to make the query $\mathsf{Corrupt}(i)$ to its challenger, which allows it to answer all queries concerning the i-th slot. □

As mentioned above, definitions 4.2 and 4.3 are equivalent. We prove the nontrivial implication below.

Proposition 4.2. *For any MCFE scheme, 1ch-IND security implies IND security.*

Proof. Let us consider an efficient MCFE adversary $\mathscr{A}$ in the IND security game. We show that $\mathscr{A}$ implies an MCFE adversary in the 1ch-IND game.

Suppose that $\mathscr{A}$ makes encryption queries for Q distinct tags during the game. The proof uses a standard hybrid argument over the distinct tags that $\mathscr{A}$ queries throughout the attack. Let H_k with $k \in \{0, 1, \ldots, Q\}$ be the game in which the challenger replies to encryption queries $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ in the following way.

- If the tag t is one of the first k distinct tags appearing in $\mathscr{A}$'s encryption queries, it replies with $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^0, t)$.
- Otherwise, it replies with $\mathsf{Encrypt}(\mathsf{ek}_i, x_i^1, t)$.

Note that an IND adversary $\mathscr{A}$ in the sense of Definition 4.2 implies a distinguisher between H_0 and H_Q.

We claim that, for any $k \in \{0, \ldots, Q-1\}$, an efficient distinguisher $\mathscr{A}_k$ between H_k and H_{k+1} implies the existence of an efficient adversary $\mathscr{B}_k$ in 1ch-IND game and such that $\mathbf{Adv}^{\text{1ch-IND}}_{\mathscr{B}_k}(\lambda) = \mathbf{Adv}^{k,k+1}_{\mathscr{A}_k}(\lambda)$. This adversary $\mathscr{B}_k$ proceeds as follows.

Initialization: Having received mpk, $\mathscr{B}_k$ runs $\mathscr{A}_k$ by feeding it with the same public parameters mpk.

Encryption queries: For each query $\mathsf{QEncrypt}(i, x_i^0, x_i^1, t)$ sent by $\mathscr{A}_k$, $\mathscr{B}_k$ does the following.

- If t is among the first k distinct tags queried by $\mathscr{A}_k$, $\mathscr{B}_k$ sends the query $\mathsf{QEncrypt}(i, x_i^0, t)$ to its own challenger in the 1ch-IND game and relays the answer back to $\mathscr{A}_k$.
- If t coincides with $(k+1)$-th distinct tag queried by $\mathscr{A}_k$, $\mathscr{B}_k$ makes the challenge query $\mathsf{QCEncrypt}(i, x_i^0, x_i^1, t)$ and transmits the response to $\mathscr{A}_k$.
- Otherwise, $\mathscr{B}_k$ sends the encryption query $\mathsf{QEncrypt}(i, x_i^1, t)$ to its challenger and forwards the answer to $\mathscr{A}_k$.

Functional decryption queries: For each query $\mathsf{QDKeygen}(f)$ made by $\mathscr{A}_k$, $\mathscr{B}_k$ sends the same query to its own challenger and passes the answer to $\mathscr{A}_k$.

Corruption queries: For each query $\mathsf{QCorrupt}(i)$ made by $\mathscr{A}_k$, $\mathscr{B}_k$ makes the same query to its own challenger and forwards the answer to $\mathscr{A}_k$.

Finalize: When $\mathscr{A}_k$ halts, $\mathscr{B}_k$ outputs the same bit b' as $\mathscr{A}_k$.

We can conclude that, for any efficient IND adversary $\mathscr{A}$, there exists an efficient 1ch-IND adversary $\mathscr{B}$ such that

$$\mathbf{Adv}_{\mathscr{A}}^{\mathrm{MCFE}}(\lambda) \leq Q \cdot \mathbf{Adv}_{\mathscr{B}}^{\mathrm{1ch-IND}}(\lambda).$$

□

4.2 MCFE for Linear Functions from KH-PRF

In this section we highlight the connection between MCFE for Inner-Products and KH-PRFs, which was implicitly used in [CDG+18a] and explicitly in [LT19] as well as [Cho19].

In the simple version of the MCFE scheme of [CDG+18a], which is itself a variant of the selectively secure Inner-Product Functional Encrypion scheme of [ABDCP15], the encryption takes place in a finite group $G = \langle g \rangle$ of prime order p, where the DDH assumption holds. In order to encrypt a message $x_i \in \mathbb{Z}_p$, under some label t, user $i \in [\ell]$, uses its secret key $s_i \in \mathbb{Z}_p$ to compute the ciphertext $C_{t,i} := g^{x_i} \cdot H(t)^{s_i} \in G$, where the function $H : \{0,1\}^* \to G$ is modeled as a random oracle. Notice that the function $F : \mathbb{Z}_p \times \{0,1\}^* \to G$ given by $F(s,t) := H(t)^s$ is the Key-Homomorphic PRF of Naor, Pinkas and Reingold [NPR00] that was discussed in Section 3.3.1. More abstractly, we can view the encryption as blinding the message x_i using the pseudo-randomness of $F(s_i, t)$, as shown in Figure 4.1 below.

This suggests the following generic construction of MCFE for linear functions using KH-PRFs as a black-box.

-	[CDG+18a]	Abstraction($F : \mathcal{K} \times \mathcal{T} \to \mathcal{M}$)
Setup :	$\{s_i \leftarrow \mathbb{Z}_p\}_{i=1}^{\ell}$, H	$\{s_i \leftarrow (\mathcal{K}, +)\}_{i=1}^{\ell}$, pp_{PRF}
Encrypt :	$C_{i,t} = g^{x_i} \cdot H(t)^{s_i}$	$C_{i,t} = x_i + F_{s_i}(t)$
DKeygen :	$SK_y = \langle \mathbf{y}, \mathbf{s} \rangle$	$SK_y = y_1 \cdot s_1 + y_2 \cdot s_2 + \cdots + y_N \cdot s_N$
Decrypt :	$\mathsf{DLog}\left(\Pi_{i=1}^{N} C_{i,t}^{y_i} / H(t)^{SK_y}\right)$	$\sum_i y_i \cdot C_{i,t} - F_{SK_y}(t)$

Figure 4.1: Parallel between a simplified [CDG+18a] variant and its abstraction

4.2.1 The Generic Construction

Given $F : \mathcal{K} \times \mathcal{T} \to \mathcal{M}$, a KH-PRF, we construct the following MCFE scheme with message space $\mathcal{M}$ and tag space $\mathcal{T}$. We will use additive notation for all the group operations done below on $\mathcal{M}$ and $\mathcal{K}$.

Setup$(1^\ell, 1^\lambda)$: Takes as input the number of users 1^ℓ and the security parameter 1^λ. Chooses a secure KH-PRF $F : \mathcal{K} \times \mathcal{T} \to \mathcal{M}$ and the secret PRF keys $\mathsf{ek}_i \in \mathcal{K}$ for each user $i \in [\ell]$. The mpk contains the description of the PRF and $\mathsf{msk} := \{\mathsf{ek}_i\}_{i \in [\ell]}$.

Encrypt(ek_i, x_i, t) : Takes as input the encryption key ek_i of user $i \in [\ell]$, a message $x_i \in \mathcal{M}$ and a tag $t \in \mathcal{T}$. It output the ciphertext $C_{t,i} := x_i + F_{\mathsf{ek}_i}(t)$.

DKeygen$(\mathsf{msk}, f_{\boldsymbol{y}})$: Takes as input the master secret key msk and an ℓ-argument function $f_{\boldsymbol{y}} : \mathcal{M}^\ell \to \mathcal{M}$, such that $f_{\boldsymbol{y}}(\boldsymbol{x}) = y_1 \cdot x_1 + y_2 + \cdot x_2 + \cdots + y_\ell \cdot x_\ell \in \mathcal{M}$, for $\boldsymbol{y} \in \mathbb{Z}^\ell$ and $\boldsymbol{x} \in \mathcal{M}^\ell$. Computes $\mathsf{ek}_{\boldsymbol{y}} = y_1 \cdot \mathsf{ek}_1 + y_2 \cdot \mathsf{ek}_2 \cdots + y_\ell \cdot \mathsf{ek}_\ell \in \mathcal{K}$ and outputs the functional decryption key $\mathsf{dk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathsf{ek}_{\boldsymbol{y}})$.

Decrypt$(\mathsf{dk}_f, t, \mathbf{C})$: Takes as input a functional decryption key $\mathsf{dk}_{\boldsymbol{y}}$, a tag t, and the ciphertexts $\mathbf{C} = (C_{t,1}, \ldots, C_{t,\ell})$. It outputs $y_1 \cdot C_{t,1} + \cdots + y_\ell \cdot C_{t,\ell} - F_{\mathsf{ek}_y}(t)$

Correctness. The algorithm Decrypt outputs the evaluation of $f_{\boldsymbol{y}}(\boldsymbol{x}) = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$ as long as all the ciphertexts are encrypted under the same label t. Indeed,

$$\sum_{i=1}^{\ell} y_i \cdot C_{t,i} = \sum_{i=1}^{\ell} (y_i \cdot x_i + y_i \cdot F_{\mathsf{ek}_i}(t)) = \langle \boldsymbol{y}, \boldsymbol{x} \rangle + F_{\mathsf{dk}_y(t)}$$

where the last equality follows from the key-homomorphic property of F.

Unfortunately, in terms of security this simple generic construction can only achieve *selective*-IND security, a weaker security model in which the adversary has to announce in advance the challenge queries in the security game of Definition 4.2. A proof for this is given in [Cho19, Theorem 4.1].

So if we want to achieve adaptive security, in the sense of Definition 4.2, we cannot rely on the pseudo-randomness of the key-homomorphic PRF in a black-box manner.

We have to mention that the simple scheme from Figure 4.1 was proved in [CDG+18a] to only be selectively secure. To achieve adaptive security they used a modified version of the fully secure functional encryption scheme of [ALS16], where they replaced the encryption randomness by random oracles. They proved adaptive security by reducing it directly to the DDH assumption.

4.3 LWE based MCFE for Linear Functions

If we plug in the LWE-based almost KH-PRF of [BLMR13] into the generic MCFE construction from the previous Section 4.2.1, we can expect a selectively-secure MCFE, based on LWE assumption, without using random oracles.

Recall that in Section 3.5 we managed to give a security proof, for the distributed version of the PRF that was adaptively secure. We did this, not by leveraging its pseudo-randomness in a modular way, but by exploiting the particularity of the construction, and reducing the security directly to the LWE assumption.

So a natural question would be: can we do a similar thing here and adapt the same KH-PRF to get an adaptively secure MCFE? The short answer is 'yes', and the approach is detailed in this section.

4.3.1 Overview

Our construction and proof for the KH-PRF component depart from the one in Section 3.4.2 in that we use an additional multiplication by $\mathbf{G}^{-1}(\mathbf{V}^\top \cdot \mathbf{G}_0)$ in order to introduce a matrix $\mathbf{V} \in \mathbb{Z}_q^{n_0 \times n}$ in the expression of $\mathbf{A}(\tau^\star)$. In addition, in this variant we do not rely on a randomness extraction argument to exploit the entropy of $\mathbf{A}(\tau^\star)^\top \cdot \mathbf{s}_i + \mathbf{e}_i$ in the challenge phase. Instead, we use a trapdoor for the matrix $\mathbf{U} = \left[\begin{smallmatrix}\mathbf{V}\\ \mathbf{C}\end{smallmatrix}\right]$ to "equivocate" the challenge ciphertexts and explain them as an encryption of $\boldsymbol{x}^\star_{1,i}$ instead of $\boldsymbol{x}^\star_{0,i}$.

The scheme encrypts $\boldsymbol{x}_i \in \mathbb{Z}^{n_0}$ as a vector $\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i$, where $\mathbf{G}_0$ is a gadget matrix; $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ is an admissible hash of the tag t; and $\mathbf{e}_i$ is a Gaussian noise. This is done in a way that a functional secret key $\mathbf{s}_y = \sum_{i=1}^{\ell} y_i \cdot \mathbf{s}_i \in \mathbb{Z}^n$ allows computing $\sum_{i=1}^{\ell} y_i \cdot \boldsymbol{x}_i$ from $\{\mathbf{C}_{t,i}\}_{i=1}^{\ell}$ by using the public trapdoor of the lattice $\Lambda^\perp(\mathbf{G}_0)$.

We derive $\mathbf{A}(\tau)$ from a set of $2L$ public matrices $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^{L}$ and an additional matrix $\mathbf{V} \in \mathbb{Z}_q^{n_0 \times n}$. Like the proof in Section 3.4.2, matrices $\mathbf{A}_{i,b} \in \mathbb{Z}_q^{n \times m}$ are interpreted as a GSW ciphertext $\mathbf{A}_{i,b} = \mathbf{A} \cdot \mathbf{R}_{i,b} + \mu_{i,b} \cdot \mathbf{G}$, where $\mathbf{R}_{i,b} \in \{-1,1\}^{m \times m}$, $\mu_{i,b} \in \{0,1\}$ and $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is the gadget matrix of [MP12]. Then, we homomorphically compute $\mathbf{A}(\tau)$ as an FHE ciphertext $\mathbf{A} \cdot \mathbf{R}'_\tau + (\prod_{i=1}^{L} \mu_{i,\tau[i]}) \cdot \mathbf{G}$, for some small-norm $\mathbf{R}'_\tau \in \mathbb{Z}^{m \times m}$, which is in turn multiplied by $\mathbf{G}^{-1}(\mathbf{V}^\top \cdot \mathbf{G}_0)$ in such a way that $\mathbf{A}(\tau) = \mathbf{A} \cdot \mathbf{R}_\tau + (\prod_{i=1}^{L} \mu_{i,\tau[i]}) \cdot (\mathbf{V}^\top \cdot \mathbf{G}_0)$. Via a careful choice of $\{\mu_{i,b}\}_{i \in [L], b \in \{0,1\}}$, the

properties of admissible hash functions imply that $\prod_{i=1}^{L} \mu_{i,x[i]}$ vanishes in all encryption queries but evaluates to 1 on the challenge tag $\tau^\star$. In order to prevent the encryption oracle from leaking too much about $\mathbf{s}_i \in \mathbb{Z}^n$, we proceed as before and replace the random $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ by a lossy matrix $\mathbf{A}^\top = \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E}$, where $\hat{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n_1\times m})$, $\mathbf{C} \hookleftarrow U(\mathbb{Z}_q^{n_1\times n})$ and for a small-norm $\mathbf{E} \in \mathbb{Z}^{m\times n}$.

4.3.2 The Construction

In the following description, we assume public parameters

$$\mathsf{cp} := \Big(\lambda,\ \ell_{\max},\ X,\ Y,\ n_0,\ n_1,\ n,\ m,\ \alpha,\ \alpha_1,\ \sigma,\ \ell_t,\ L,\ q,\ \mathsf{AHF}\Big),$$

consisting of a security parameter λ and the following quantities:

- $(X, Y, \ell_{\max}, n_0, n_1, n, m)$, which are all in $\mathsf{poly}(\lambda)$

  $X = 1$, $n_1 = \lambda^d$, $q = 2^{\lambda^{d-1}}$, $\alpha = 2^{-\sqrt{\lambda}}$, $\alpha_1 = 2^{-\lambda^{d-1}+d\log\lambda}$, $n_0 = o(\lambda^{d-2})$, $n = O(\lambda^{2d-1})$, $\sigma = 2^{\lambda^{d-1}-2\lambda}$ and $n_0 \cdot \ell_{max} = O(\lambda^{d-2})$ where d is a constant; for instance $d = 3$ works asymptotically.

- The description of a tag space $\mathcal{T} = \{0,1\}^{\ell_t}$, for some $\ell_t \in \mathsf{poly}(\lambda)$, such that tags may be arbitrary strings (e.g., time period numbers or dataset names).

- A balanced admissible hash function $\mathsf{AHF} : \{0,1\}^{\ell_t} \to \{0,1\}^L$, for a suitable $L \in \Theta(\lambda)$ (Def. 2.1).

- The message space will be $\mathcal{M} = [-X, X]^{n_0}$, for some $n_0 \in \mathsf{poly}(\lambda)$.

- Integers $n, n_0, n_1, m \in \mathsf{poly}(\lambda)$ satisfying the conditions $m > 2n \cdot \lceil \log q \rceil$ and $n > 3 \cdot (n_0 + n_1) \cdot \lceil \log q \rceil$.

- A real $\alpha > 0$ and a Gaussian parameter $\sigma > 0$, which specifies an interval $[-\beta, \beta] = [-\sigma\sqrt{n}, \sigma\sqrt{n}]$ where the coordinates of users' secret keys will live (with probability exponentially close to 1).

Letting $\ell \in \mathsf{poly}(\lambda)$, with $\ell \le \ell_{max}$, be the number of users, our function space is the set of all functions $f_{\boldsymbol{y}} : \mathbb{Z}^{n_0\times\ell} \to \mathbb{Z}^{n_0}$ indexed by an integer vector $\boldsymbol{y} \in \mathbb{Z}^\ell$ of infinity norm $\|\boldsymbol{y}\|_\infty < Y$.

We define $\mathbf{G}_0 \in \mathbb{Z}_q^{n_0\times m}$ to be the gadget matrix

$$\mathbf{G}_0 = [\mathbf{I}_{n_0} \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil}) \mid \mathbf{0}^{n_0} \mid \ldots \mid \mathbf{0}^{n_0}] \in \mathbb{Z}_q^{n_0\times m}$$

where the product $\mathbf{I}_{n_0} \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil})$ is padded with $m - n_0 \cdot \lceil \log q \rceil$ zero columns. We similarly denote by $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ the gadget matrix of rank n:

$$\mathbf{G} = [\mathbf{I}_n \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil}) \mid \mathbf{0}^n \mid \ldots \mid \mathbf{0}^n] \in \mathbb{Z}_q^{n\times m}.$$

Our MCFE construction goes as follows.

Setup$(\mathsf{cp}, 1^\ell)$**:** On input of cp and a number of users ℓ, do the following.

1. Choose random matrices $\mathbf{A}_{i,j} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, for each $i \in [L]$, $j \in \{0,1\}$.
2. Choose a uniformly random matrix $\mathbf{V} \hookleftarrow U(\mathbb{Z}_q^{n_0\times n})$.
3. For each $i \in [\ell]$, sample $\mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ and define $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$.

Output the master secret key $\mathsf{msk} := \{\mathsf{ek}_i\}_{i=1}^\ell$ and the public parameters

$$\mathsf{mpk} := \Big(\mathsf{cp},\ \mathbf{V},\ \{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\ \in \mathbb{Z}_q^{n\times m}\}_{i=1}^L\Big).$$

DKeygen$(\mathsf{msk}, f_{\boldsymbol{y}})$: Given the master secret key $\mathsf{msk} := \{\mathsf{ek}_i\}_{i=1}^\ell$ and a linear function $f_{\boldsymbol{y}} : \mathbb{Z}^{n_0\times\ell} \to \mathbb{Z}^{n_0}$ defined by an integer vector $\boldsymbol{y} = (y_1,\ldots,y_\ell)^\top \in \mathbb{Z}^\ell$ which maps an input $\mathbf{X} = [\boldsymbol{x}_1 \mid \ldots \mid \boldsymbol{x}_\ell] \in \mathbb{Z}^{n_0\times\ell}$ to $f_{\boldsymbol{y}}(\mathbf{X}) = \mathbf{X}\cdot\boldsymbol{y} \in \mathbb{Z}^{n_0}$, parse each ek_i as a vector $\mathbf{s}_i \in \mathbb{Z}^n$. Then, compute and output the functional secret key $\mathsf{dk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathbf{s}_{\boldsymbol{y}})$, where $\mathbf{s}_{\boldsymbol{y}} = \sum_{i=1}^\ell \mathbf{s}_i \cdot y_i \in \mathbb{Z}^n$.

Encrypt$(\mathsf{ek}_i, \boldsymbol{x}_i, t)$: Given $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^n$, $\boldsymbol{x}_i \in [-X, X]^{n_0}$, and $t \in \{0,1\}^{\ell_t}$,

1. Compute $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ and parse it as $\tau = \tau_1\ldots\tau_L$.
2. Define $\mathbf{W} = \mathbf{G}_0^\top \cdot \mathbf{V} \in \mathbb{Z}_q^{m\times n}$ and compute

$$\mathbf{A}(\tau) = \mathbf{A}_{1,\tau_1}\cdot\mathbf{G}^{-1}\Big(\mathbf{A}_{2,\tau_2}\cdot\mathbf{G}^{-1}\big(\ldots\mathbf{A}_{L-1,\tau_{L-1}}\cdot\mathbf{G}^{-1}\big(\mathbf{A}_{L,\tau_L}\big)\big)\Big) \cdot\mathbf{G}^{-1}(\mathbf{W}^\top)\ \in \mathbb{Z}_q^{n\times m}. \tag{4.1}$$

3. Sample a noise vector $\mathbf{e}_i \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$. Then, compute and output

$$\mathbf{C}_{t,i} = \mathbf{G}_0^\top\cdot\boldsymbol{x}_i + \mathbf{A}(\tau)^\top\cdot\mathbf{s}_i + \mathbf{e}_i\ \in \mathbb{Z}_q^m.$$

Decrypt$(\mathsf{dk}_{\boldsymbol{y}}, t, \mathbf{C}_t)$: On input of a functional secret key $\mathsf{dk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{s}_{\boldsymbol{y}})$ for a vector $\boldsymbol{y} = (y_1,\ldots,y_\ell)^\top \in [-Y, Y]^\ell$, a tag $t \in \{0,1\}^{\ell_t}$, and an ℓ-vector of ciphertexts $\mathbf{C}_t = (\mathbf{C}_{t,1},\ldots,\mathbf{C}_{t,\ell}) \in (\mathbb{Z}_q^m)^\ell$, conduct the following steps.

1. Compute $\tau = \mathsf{AHF}(t) \in \{0,1\}^L$ and parse it as $\tau = \tau_1\ldots\tau_L$.
2. Compute $\mathbf{A}(\tau) \in \mathbb{Z}_q^{n\times m}$ as per (4.1).
3. Compute $\mathbf{f}_{t,\boldsymbol{y}} = \sum_{i=1}^\ell y_i\cdot\mathbf{C}_{t,i} - \mathbf{A}(\tau)^\top\cdot\mathbf{s}_{\boldsymbol{y}} \mod q$.
4. Interpret $\mathbf{f}_{t,\boldsymbol{y}} \in \mathbb{Z}_q^m$ as a vector of the form $\mathbf{f}_{t,\boldsymbol{y}} = \mathbf{G}_0^\top\cdot\boldsymbol{z} + \tilde{\mathbf{e}} \mod q$, for some "small" error vector $\tilde{\mathbf{e}}$. Using the public trapdoor of $\Lambda^\perp(\mathbf{G}_0)$, compute and output the underlying vector $\boldsymbol{z} \in [-\ell\cdot X\cdot Y, \ell\cdot X\cdot Y]^{n_0}$.

4.3.3 Correctness and Security

Lemma 4.1 (Correctness)**.** *Assume that* $\alpha q = \omega(\sqrt{\log \ell})$*,* $Y \cdot \ell \cdot \alpha q \cdot \log q < q/2$ *and* $\ell \cdot X \cdot Y < q/2$*. Then, for any* $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(\mathsf{cp}, 1^\lambda)$*, any message* $\mathbf{X} = [\boldsymbol{x}_1 | \cdots | \boldsymbol{x}_\ell] \in [-X, X]^{n_0 \times \ell}$*, any* $\boldsymbol{y} \in [-Y, Y]^\ell$*, any tag* $t \in \{0,1\}^{\ell_t}$*, algorithm* $\mathsf{Decrypt}(\mathsf{dk}_{\boldsymbol{y}}, t, \mathbf{C}_t)$ *outputs* $\mathbf{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$ *with probability exponentially close to* 1*, where* $\mathbf{C}_{t,i} \leftarrow \mathsf{Encrypt}(\mathsf{ek}_i, \boldsymbol{x_i}, t)$ *and* $\mathsf{dk}_y \leftarrow \mathsf{DKeygen}(\mathsf{msk}, f_{\boldsymbol{y}})$*.*

Proof. Suppose that $\mathbf{C}_{t,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i + \mathbf{A}(\tau)^\top \cdot \mathbf{s}_i + \mathbf{e}_i \in \mathbb{Z}_q^m$, with each error vector $\boldsymbol{e}_i \leftarrow D_{\mathbb{Z}^m, \alpha q}$ and $\tau = \mathsf{AHF}(t)$. Defining $\mathbf{E} = [\boldsymbol{e}_1 | \cdots | \boldsymbol{e}_\ell] \in \mathbb{Z}^{m \times \ell}$, we have

$$\mathbf{f}_{t,\boldsymbol{y}} = \sum_{i=1}^{\ell} y_i \cdot \mathbf{C}_{t,i} - \mathbf{A}(\tau)^\top \cdot \mathbf{s}_{\boldsymbol{y}} = \mathbf{G}_0^\top \cdot \mathbf{X} \cdot \boldsymbol{y} + \mathbf{E} \cdot \boldsymbol{y} \mod q.$$

We know from [MP12] that $\Lambda^\perp(\mathbf{G}_0)$ has a public trapdoor $\mathbf{T}_0 \in \mathbb{Z}^{m \times m}$ such that $\mathbf{G}_0 \cdot \mathbf{T}_0 = \mathbf{0} \mod q$ and $\|\mathbf{T}_0^\top\|_\infty \leq \log q$ where $\|\mathbf{T}_0^\top\|_\infty := \sup_{\boldsymbol{x} \neq \mathbf{0}} \frac{\|\mathbf{T}_0^\top \cdot \boldsymbol{x}\|_\infty}{\|\boldsymbol{x}\|_\infty}$. By using this short trapdoor, we are able to recover $\mathbf{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$ in the following way: observe that $\mathbf{T}_0^\top \cdot \mathbf{f}_{t,\boldsymbol{y}} = \mathbf{T}_0^\top \cdot \mathbf{E} \cdot \boldsymbol{y} \mod q$. By [MR07, Lemma 4.4], we can bound the Euclidean norm of a Gaussian vector $\mathbf{e} \hookleftarrow D_{\mathbb{Z}^\ell, \alpha q}$ by $\|\mathbf{e}\| \leq \sqrt{\ell} \cdot \alpha q$ with probability exponentially close to 1. It comes that

$$\|\mathbf{E}\|_\infty = \max_{i \in [m]} \sum_{j=1}^{\ell} |e_{i,j}| \leq \sqrt{\ell} \cdot \max_{i \in [m]} \sqrt{\sum_{j=1}^{\ell} e_{i,j}^2} \leq \ell \cdot \alpha q.$$

Moreover, the hypotheses imply

$$\|\mathbf{T}_0^\top \cdot \mathbf{E} \cdot \boldsymbol{y}\|_\infty \leq \|\mathbf{T}_0^\top\|_\infty \cdot \|\mathbf{E}\|_\infty \cdot \|\boldsymbol{y}\|_\infty \leq \log q \cdot \ell \cdot \alpha q \cdot Y < q/2.$$

Hence, $\mathbf{T}_0^\top \cdot \mathbf{f}_{t,\boldsymbol{y}} \mod q$ actually reveals $\mathbf{T}_0^\top \cdot \mathbf{E} \cdot \boldsymbol{y}$ over $\mathbb{Z}^m$. Since $\mathbf{T}_0^\top$ is a $\mathbb{Z}$-basis, it comes that we can recover $\mathbf{E} \cdot \boldsymbol{y} \in \mathbb{Z}^m$, at which point we also get $\mathbf{X} \cdot \boldsymbol{y} \mod q$. Since $\|\mathbf{X} \cdot \boldsymbol{y}\|_\infty \leq \ell \cdot X \cdot Y < q/2$ by hypothesis, the modular product $\mathbf{X} \cdot \boldsymbol{y} \mod q$ is nothing but $\mathbf{X} \cdot \boldsymbol{y} \in \mathbb{Z}^{n_0}$. □

We now prove the security of the scheme in the sense of Definition 4.3 (and thus Definition 4.2 modulo some loss of tightness in the reduction).

For the current parameters $n_1 = \lambda^d$, $q = 2^{\lambda^{d-1}}$, and $\alpha_1 = 2^{-\lambda^{d-1} + d \log \lambda}$, $\alpha_1 q = \Omega(\sqrt{n_1})$, we know from [Reg05] that $\mathsf{LWE}_{q, n_1, \alpha_1}$ is at least as hard as GapSVP_γ, with $\gamma = \tilde{O}(n_1 / \alpha_1) = \tilde{O}(2^{\lambda^{d-1}})$. The best known algorithms [Sch87] for solving GapSVP_γ run in $2^{\tilde{O}\left(\frac{n_1}{\log \gamma}\right)}$, which for our parameters is $2^{\tilde{O}(\lambda)}$.

Theorem 4.2. *Under the* $\mathsf{LWE}_{q, m, n_1, \alpha_1}$ *assumption, the above MCFE scheme provides adaptive security.*

Proof. The proof considers a sequence of games. In each game, we denote by W_i the event that $b' = b$. For each i, the adversary's advantage function in Game_i is $\mathbf{Adv}_{\mathscr{A}}^i(\lambda) := |\Pr[b' = b] - 1/2| = \frac{1}{2} \cdot |\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]|$.

Game$_0$: This is the real security game. We denote by $t^\star$ the tag of the challenge phase while $t^{(1)}, \ldots, t^{(Q)}$ are the tags involved in encryption queries. Namely, for each $j \in [Q]$, $t^{(j)}$ stands for the j-th distinct tag involved in an encryption query. Since up to ℓ encryption queries $(i, \boldsymbol{x}_i, t)$ are allowed for each tag t, the adversary can make a total of $\ell \cdot Q$ encryption queries. The game begins with the challenger initially choosing encryption keys $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ by sampling $\mathsf{ek}_i = \mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ for each $i \in [\ell]$. In addition, the challenger flips a fair coin $b \hookleftarrow U(\{0,1\})$ which will determine the response to challenge queries. At each corruption query $i \in [\ell]$, the adversary obtains ek_i and the challenger updates a set $\mathscr{CS} := \mathscr{CS} \cup \{i\}$, which is initially empty. At each encryption query $(i, \boldsymbol{x}_i^{(j)}, t^{(j)})$, the challenger samples $\mathbf{e}_i^{(j)} \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ and returns

$$\mathbf{C}_{t,i}^{(j)} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i^{(j)} + \mathbf{A}(\tau^{(j)})^\top \cdot \mathbf{s}_i + \mathbf{e}_i^{(j)} \in \mathbb{Z}_q^m,$$

where $\tau^{(j)} = \mathsf{AHF}(t^{(j)})$. In the challenge phase, the adversary $\mathscr{A}$ chooses a fresh tag $t^\star$ and two vectors of messages $\mathbf{X}_0^\star = [\boldsymbol{x}_{0,1}^\star \mid \ldots \mid \boldsymbol{x}_{0,\ell}^\star] \in [-X, X]^{n_0 \times \ell}$ and $\mathbf{X}_1^\star = [\boldsymbol{x}_{1,1}^\star \mid \ldots \mid \boldsymbol{x}_{1,\ell}^\star] \in [-X, X]^{n_0 \times \ell}$ subject to the constraint that, for any private key query $\boldsymbol{y} \in [-Y, Y]^\ell$ made by $\mathscr{A}$, we must have $\mathbf{X}_0^\star \cdot \boldsymbol{y} = \mathbf{X}_1^\star \cdot \boldsymbol{y}$ over $\mathbb{Z}$. In addition, the invariant that $\boldsymbol{x}_{0,i}^\star = \boldsymbol{x}_{1,i}^\star$ for any $i \in \mathscr{CS}$ must be satisfied at any time during the game. In response to a challenge query $(i, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star, t^\star)$, the challenger generates a challenge ciphertext $\mathbf{C}_{t^\star,i}$, where

$$\mathbf{C}_{t^\star,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{b,i}^\star + \mathbf{A}(\tau^\star)^\top \cdot \mathbf{s}_i + \mathbf{e}_i^\star, \tag{4.2}$$

where $\tau^\star = \mathsf{AHF}(t^\star)$ and $\mathbf{e}_i^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ for all $i \in [\ell]$.
When $\mathscr{A}$ halts, it outputs $\hat{b} \in \{0,1\}$ and the challenger defines $b' := \hat{b}$. We have $\mathbf{Adv}_{\mathscr{A}}(\lambda) := |\Pr[W_0] - 1/2|$, where W_0 is event that $b' = b$.

Game$_1$: This game is identical to Game$_0$ except for the following changes. The challenger runs $K \leftarrow \mathsf{AdmSmp}(1^\lambda, Q_{\max}, \delta)$ to generate a key $K \in \{0,1,\perp\}^L$ for a balanced admissible hash function $\mathsf{AHF} : \{0,1\}^{\ell_t} \to \{0,1\}^L$, where $\delta := \mathbf{Adv}_{\mathscr{A}}(\lambda)$ and $Q_{\max}$ is the maximum number of distinct tags that the adversary queries during the game. When the adversary halts and outputs $\hat{b} \in \{0,1\}$, the challenger checks if the conditions

$$P_K(t^{(1)}) = \cdots = P_K(t^{(Q)}) = 1 \ \wedge \ P_K(t^\star) = 0 \tag{4.3}$$

are satisfied. If conditions (4.3) do not hold, the challenger ignores $\mathscr{A}$'s output $\hat{b} \in \{0,1\}$ and overwrites it with a random bit $b'' \hookleftarrow \{0,1\}$ to define $b' = b''$. If conditions (4.3) are satisfied, the challenger sets $b' = \hat{b}$. By Lemma 2.17,

$$|\Pr[W_1] - 1/2| \quad \geq \quad \gamma_{\min} \cdot \mathbf{Adv}_{\mathscr{A}}(\lambda) - \frac{1}{2} \cdot (\gamma_{\max} - \gamma_{\min}),$$

where the right hand side is a noticeable function in λ.

Game$_2$: In this game, we modify the generation of mpk in the following way. Initially, the challenger samples a uniformly random matrix $\mathbf{A} \leftarrow U(\mathbb{Z}_q^{n\times m})$. Next, for each $i \in [L]$, it samples $\mathbf{R}_{i,0}, \mathbf{R}_{i,1} \leftarrow U(\{-1,1\})^{m\times m}$ and defines the public matrices $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^{L}$ as follows: for all $i \in [L]$ and $j \in \{0,1\}$

$$\mathbf{A}_{i,j} := \begin{cases} \mathbf{A}\cdot\mathbf{R}_{i,j} & \text{if } (j \neq K_i) \wedge (K_i \neq \perp) \\ \mathbf{A}\cdot\mathbf{R}_{i,j} + \mathbf{G} & \text{if } (j = K_i) \vee (K_i = \perp) \end{cases} \tag{4.4}$$

Since $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ was chosen uniformly, the Leftover Hash Lemma ensures that $\{\mathbf{A}_{i,0}, \mathbf{A}_{i,1}\}_{i=1}^{L}$ are statistically independent and uniformly distributed over $\mathbb{Z}_q^{n\times m}$. It follows that $|\Pr[W_2] - \Pr[W_1]| \leq L \cdot 2^{-\lambda}$.

We note that, at each encryption query $(i, \boldsymbol{x}_i^{(j)}, t^{(j)})$, the admissible hash function maps $t^{(j)}$ to $\tau^{(j)} = \mathsf{AHF}(t^{(j)})$, which is itself mapped to a GSW encryption

$$\mathbf{A}(\tau^{(j)}) = \mathbf{A}\cdot\mathbf{R}_{\tau^{(j)}} + (\prod_{i=1}^{L} \mu_i)\cdot\mathbf{W}^\top, \tag{4.5}$$

of a product $\prod_{i=1}^{L}\mu_i$, for some small norm matrix $\mathbf{R}_{\tau^{(j)}} \in \mathbb{Z}^{m\times m}$, where

$$\mu_i := \begin{cases} 0 & \text{if } (\mathsf{AHF}(t^{(j)})_i \neq K_i) \wedge (K_i \neq \perp) \\ 1 & \text{if } (\mathsf{AHF}(t^{(j)})_i = K_i) \vee (K_i = \perp) \end{cases}$$

If conditions (4.3) are satisfied, at each encryption query $(i, x_i^{(j)}, t^{(j)})$, the admissible hash function ensures that $\tau^{(j)} = \mathsf{AHF}(t^{(j)})$ satisfies

$$\mathbf{A}(\tau^{(j)}) = \mathbf{A}\cdot\mathbf{R}_{\tau^{(j)}} \qquad \forall\, j \in [Q], \tag{4.6}$$

for some small norm $\mathbf{R}_{\tau^{(j)}} \in \mathbb{Z}^{m\times m}$. Moreover, the challenge tag $t^\star$ is mapped to an L-bit string $\tau^\star = \mathsf{AHF}(t^\star)$ such that

$$\mathbf{A}(\tau^\star) \;=\; \mathbf{A}\cdot\mathbf{R}_{\tau^\star} + \mathbf{W}^\top = \mathbf{A}\cdot\mathbf{R}_{\tau^\star} + \mathbf{V}^\top\cdot\mathbf{G}_0 \tag{4.7}$$

Game$_3$: In this game, we modify the distribution of mpk and replace the uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ by a lossy matrix such that

$$\mathbf{A}^\top = \hat{\mathbf{A}}^\top\cdot\mathbf{C} + \mathbf{E} \in \mathbb{Z}_q^{m\times n}, \tag{4.8}$$

where $\hat{\mathbf{A}} \leftarrow U(\mathbb{Z}_q^{n_1\times m})$, $\mathbf{C} \leftarrow U(\mathbb{Z}_q^{n_1\times n})$ and $\mathbf{E} \leftarrow D_{\mathbb{Z}^{m\times n},\alpha_1 q}$, for $n_1 \ll n$. The matrix (4.8) is thus "close" to a matrix $\hat{\mathbf{A}}^\top\cdot\mathbf{C}$ of much lower rank than n. Under the LWE assumption in dimension n_1 with error rate α_1, this change should not significantly affect $\mathscr{A}$'s behavior and a straightforward reduction $\mathscr{B}$ shows that $|\Pr[W_3] - \Pr[W_2]| \leq n \cdot \mathbf{Adv}^{\mathsf{LWE}_{q,m,n_1,\alpha_1}}_{\mathscr{B}}(\lambda)$, where the factor n comes from the use of an LWE assumption with n secrets.

Game$_4$: In this game, we modify the encryption oracle. At each encryption query $(i, \boldsymbol{x}_i^{(j)}, t^{(j)})$, the challenger generates the ciphertext by computing:

$$\mathbf{C}_{t,i}^{(j)} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_i^{(j)} + \mathbf{R}_{\tau^{(j)}}^\top \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} \cdot \mathbf{s}_i + \mathbf{e}_i^{(j)} \in \mathbb{Z}_q^m, \tag{4.9}$$

and for each challenge query $(i, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star, t^\star)$ the challenger replies with:

$$\mathbf{C}_{t^\star,i} = \mathbf{G}_0^\top \cdot \boldsymbol{x}_{b,i}^\star + \left(\mathbf{R}_{\tau^\star}^\top \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{G}_0^\top \cdot \mathbf{V}\right) \cdot \mathbf{s}_i + \mathbf{e}_i^\star \in \mathbb{Z}_q^m \tag{4.10}$$

where $\mathbf{e}_i^{(j)} \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ and $\mathbf{e}_i^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$. The only difference between Game$_3$ and Game$_4$ is thus that the terms $\mathbf{R}_{\tau^{(j)}}^\top \cdot \mathbf{E} \cdot \mathbf{s}_i + \mathbf{e}_i^{(j)}$ and $\mathbf{R}_{\tau^\star}^\top \cdot \mathbf{E} \cdot \mathbf{s}_i + \mathbf{e}_i^\star$ are replaced by $\mathbf{e}_i^{(j)}$ and $\mathbf{e}_i^\star$ respectively, at each encryption or challenge query. However, the smudging lemma (Lemma 2.9) ensures that the two distributions are statistically close as long as α is sufficiently large with respect to α_1 and σ. Concretely, Lemma 4.3 implies $|\Pr[W_4] - \Pr[W_3]| \leq \ell \cdot (Q+1) \cdot 2^{-\Omega(\lambda)}$.

Game$_5$: This game is like Game$_4$ but we modify the challenge oracle. Instead of encrypting $\mathbf{X}_b^\star = [\boldsymbol{x}_{b,1}^\star \mid \ldots \mid \boldsymbol{x}_{b,\ell}^\star]$ as in (4.10), the challenger encrypts a linear combination of $\mathbf{X}_0^\star$ and $\mathbf{X}_1^\star$. It initially chooses a uniformly random $\gamma \hookleftarrow U(\mathbb{Z}_q)$ and, at each challenge query $(i, \boldsymbol{x}_{0,i}^\star, \boldsymbol{x}_{1,i}^\star, t^\star)$, computes $\mathbf{C}_{t^\star,i}$ as

$$\mathbf{C}_{t^\star,i} = \mathbf{G}_0^\top \cdot \left((1-\gamma) \cdot \boldsymbol{x}_{b,i}^\star + \gamma \cdot \boldsymbol{x}_{1-b,i}^\star\right) + \left(\mathbf{R}_{\tau^\star}^\top \cdot \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{G}_0^\top \cdot \mathbf{V}\right) \cdot \mathbf{s}_i + \mathbf{e}_i^\star,$$

with $\mathbf{e}_i^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$, for all $i \in [\ell]$. Lemma 4.4 shows that Game$_4$ and Game$_5$ are negligibly far part as $|\Pr[W_5] - \Pr[W_4]| \leq 2^{-\Omega(\lambda)}$.

In Game$_5$, we have $\Pr[W_5] = 1/2$ since the challenge ciphertexts $(\mathbf{C}_{t,1}^\star, \ldots, \mathbf{C}_{t,\ell}^\star)$ reveal no information about $b \in \{0,1\}$. □

Lemma 4.3. *Let $\mathbf{R}_\tau \in \mathbb{Z}^{m \times m}$ be as in equation (4.5). Let $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m \times n},\alpha_1 q}$ and $\mathbf{s} \hookleftarrow D_{\mathbb{Z}^n,\sigma}$. If $\alpha_1 q = \omega(\sqrt{\log n})$, $\sigma = \omega(\sqrt{\log n})$ and $\alpha \geq 2^\lambda \cdot L \cdot m^4 \cdot n^{3/2} \cdot \alpha_1 \cdot \sigma$, we have the statistical distance upper bound $\Delta\left(D_{\mathbb{Z}^m,\alpha q},\ \mathbf{R}_\tau^\top \cdot \mathbf{E} \cdot \mathbf{s} + D_{\mathbb{Z}^m,\alpha q}\right) \leq 2^{-\lambda}$.*

Proof. If the matrices $\mathbf{A}_{i,j} = \mathbf{A} \cdot \mathbf{R}_{i,j} + \mu_{i,j} \cdot \mathbf{G} \in \mathbb{Z}^{n \times m}$, for $i \in [L]$ and $j \in \{0,1\}$, with $\mathbf{R}_{i,j} \hookleftarrow \{-1,1\}^{m \times m}$ and $\mu_{i,j} \in \{0,1\}$, defined as in equation (4.4), then $\mathbf{R}_\tau = \mathbf{R}'_\tau \cdot \mathbf{G}^{-1}(\mathbf{W}^\top)$, where $\mathbf{R}'_\tau$ is given below:

$$\begin{aligned}
\mathbf{R}'_\tau \quad = \quad & \mathbf{R}_{1,\tau_1} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{2,\tau_2} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{3,\tau_3} \cdot \left(\cdots \mathbf{G}^{-1}\left(\mathbf{A}_{L,\tau_L}\right)\right)\right)\right) \\
& + \tau_1 \cdot \mathbf{R}_2 \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{3,\tau_3} \cdot \left(\cdots \mathbf{G}^{-1}\left(\mathbf{A}_{L,\tau_L}\right)\right)\right) \\
& + x_1 \cdot x_2 \cdot \mathbf{R}_3 \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{4,\tau_4} \cdot \left(\cdots \mathbf{G}^{-1}\left(\mathbf{A}_{L,\tau_L}\right)\right)\right) \\
& \vdots \\
& + \tau_1 \cdot \tau_2 \cdots \tau_{L-2} \cdot \mathbf{R}_{L-1} \cdot \mathbf{G}^{-1}\left(\mathbf{A}_{L,\tau_L}\right) \\
& + \tau_1 \cdot \tau_2 \cdots \tau_{L-1} \cdot \mathbf{R}_L
\end{aligned}$$

$\mathbf{R}_\tau^\top$ is thus a sum of L terms of products of at most 3 binary matrices. Since each binary matrix has infinity norm less than m we obtain the upper bound $\|\mathbf{R}_\tau^\top\|_\infty \leq L \cdot m^3$. By [MR07, Lemma 4.4], we can bound the Euclidean norm of a Gaussian vector $\boldsymbol{e} \hookleftarrow D_{\mathbb{Z}^n,\alpha_1 q}$ by $\|\mathbf{e}\| \leq \sqrt{n}\cdot\alpha_1 q$, with probability exponentially close to 1. We thus obtain

$$\|\mathbf{E}\|_\infty = \max_{i\in[m]} \sum_{j=1}^{n} |e_{i,j}| \leq \sqrt{n}\cdot \max_{i\in[m]} \sqrt{\sum_{j=1}^{n} e_{i,j}^2} \leq n\cdot\alpha_1 q,$$

so that $\|\mathbf{R}_\tau^\top \cdot \mathbf{E}\cdot\mathbf{s}\|_\infty \leq \|\mathbf{R}_\tau^\top\|_\infty \cdot \|\mathbf{E}\|_\infty \cdot \|\mathbf{s}\|_\infty \leq Lm^3 \cdot n\alpha_1 q \cdot \sigma\sqrt{n}$. By Lemma 2.9, the statistical distance to be bounded is at most $\leq m \cdot \frac{Lm^3\cdot n\alpha_1 q\cdot\sigma\sqrt{n}}{\alpha q} = 2^{-\lambda}$. □

Lemma 4.4. *We have* $|\Pr[W_5] - \Pr[W_4]| \leq 2^{-\Omega(\lambda)}$.

Proof. To prove the result, we resort to a technique of guessing in advance the difference $\mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$, which was previously used in [Wee14, BBL17] and can be seen as complexity leveraging with respect to a statistical argument. We consider the following variants of Game_4 and Game_5, respectively.

We define Game'_4 and Game'_5 simultaneously by using an index $k \in \{4,5\}$:

Game'_k: This game is like Game_k with one difference in the setup phase. To generate mpk, the challenger generates a statistically uniform $\mathbf{U} \in \mathbb{Z}_q^{(n_0+n_1)\times n}$ with a trapdoor $\mathbf{T}_U$ (Lemma 2.8) for the lattice $\Lambda^\perp(\mathbf{U})$. Then, $\mathscr{B}$ parses $\mathbf{U}$ as

$$\mathbf{U} = \begin{bmatrix}\mathbf{V}\\ \mathbf{C}\end{bmatrix} \in \mathbb{Z}_q^{(n_0+n_1)\times n},$$

where $\mathbf{V} \in \mathbb{Z}_q^{n_0\times n}$ and $\mathbf{C} \in \mathbb{Z}_q^{n_1\times n}$ are statistically independent and uniform over $\mathbb{Z}_q$. Next, it computes

$$\mathbf{A}^\top = \hat{\mathbf{A}}^\top \cdot \mathbf{C} + \mathbf{E} \in \mathbb{Z}_q^{m\times n},$$

where $\hat{\mathbf{A}} \hookleftarrow U(\mathbb{Z}_q^{n_1\times m})$ and $\mathbf{E} \hookleftarrow D_{\mathbb{Z}^{m\times n},\alpha_1 q}$. The obtained matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is then used to generate $\{\mathbf{A}_{i,j}\}_{i\in[L],j\in\{0,1\}}$ as per (4.4). The upper part $\mathbf{V} \in \mathbb{Z}_q^{n_0\times n}$ of $\mathbf{U}$ is included in mpk, the distribution of which is statistically close to that of Game_k: we indeed have $|\Pr[W'_k] - \Pr[W_k]| \leq 2^{-\Omega(\lambda)}$.

We do the same as above and define Game''_4 and Game''_5 simultaneously by using an index $k \in \{4,5\}$:

Game''_k: This game is identical to Game'_k with the following difference. At the outset of the game, the challenger randomly chooses $\Delta\mathbf{X} \hookleftarrow U([-2X,2X]^{n_0\times\ell})$ as a guess for the difference $\mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$ between the challenge messages $\mathbf{X}^\star_0, \mathbf{X}^\star_1$. In the challenge phase, the challenger checks if $\Delta\mathbf{X} = \mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$. If not, it aborts and replaces $\mathscr{A}$'s output $\hat{b}$ with a random bit $b'' \hookleftarrow U(\{0,1\})$. If the guess for $\mathbf{X}^\star_{1-b} - \mathbf{X}^\star_b$ was successful (we call Guess this event), the challenger proceeds exactly as it did in Game'_k.

Since the choice of $\Delta\mathbf{X} \hookleftarrow U([-2X,2X]^{n_0\times\ell})$ is completely independent of $\mathscr{A}$'s view, we clearly have $\Pr[\mathsf{Guess}] = 1/(4X)^{n_0\ell}$. Since Game_4'' is identical to Game_4' when Guess occurs, this implies $\mathbf{Adv}^{4'}_{\mathscr{A}}(\lambda) = (4X)^{n_0\ell}\cdot\mathbf{Adv}^{4''}_{\mathscr{A}}(\lambda)$. Indeed,

$$\begin{aligned}\mathbf{Adv}^{4''}_{\mathscr{A}}(\lambda) &:= \frac{1}{2}\cdot|\Pr[b'=1\mid b=1,\mathsf{Guess}]\cdot\Pr[\mathsf{Guess}]+\frac{1}{2}\cdot\Pr[\neg\mathsf{Guess}]\\ &\qquad -\Pr[b'=1\mid b=0,\mathsf{Guess}]\cdot\Pr[\mathsf{Guess}]-\frac{1}{2}\cdot\Pr[\neg\mathsf{Guess}]|\\ &= \frac{1}{2}\cdot\Pr[\mathsf{Guess}]\cdot|\Pr[b'=1\mid b=1,\mathsf{Guess}]-\Pr[b'=1\mid b=0,\mathsf{Guess}]|\\ &= \Pr[\mathsf{Guess}]\cdot\mathbf{Adv}^{4'}_{\mathscr{A}}(\lambda)=\frac{1}{(4X)^{n_0\ell}}\cdot\mathbf{Adv}^{4'}_{\mathscr{A}}(\lambda)\end{aligned}$$

and we can similarly show that $\mathbf{Adv}^{5'}_{\mathscr{A}}(\lambda) = (4X)^{n_0\ell}\cdot\mathbf{Adv}^{5''}_{\mathscr{A}}(\lambda)$.

Game_5''': This game is identical to Game_4'' except that encryption keys $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ are replaced by alternative encryption keys $\{\mathsf{ek}'_i\}_{i=1}^{\ell}$, which are generated as follows. After having sampled $\mathsf{ek}_i = \mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^n,\sigma}$ for all $i\in[\ell]$, the challenger chooses $\gamma\hookleftarrow U(\mathbb{Z}_q)$ and uses the trapdoor $\mathbf{T_U}$ for $\Lambda^{\perp}(\mathbf{U})$ to sample a small-norm matrix $\mathbf{T}\in\mathbb{Z}^{n\times n_0}$ (Lemma 2.8, Remark 1) satisfying

$$\mathbf{U}\cdot\mathbf{T} = \left[\frac{\gamma\cdot\mathbf{I}_{n_0}}{\mathbf{0}^{n_1\times n_0}}\right] \bmod q, \tag{4.11}$$

so that $\mathbf{V}\cdot\mathbf{T} = \gamma\cdot\mathbf{I}_{n_0} \bmod q$ and $\mathbf{C}\cdot\mathbf{T} = \mathbf{0}^{n_1\times n_0} \bmod q$. For each $i\in[\ell]$, the alternative key $\mathsf{ek}'_i = \mathbf{s}'_i$ of user i is defined as:

$$\mathbf{s}'_i = \mathbf{s}_i + \mathbf{T}\cdot\Delta\boldsymbol{x}_i \ \in\mathbb{Z}^n \qquad \forall i\in[\ell], \tag{4.12}$$

where $\Delta\boldsymbol{x}_i$ is the i-th column of $\Delta\mathbf{X}$ (i.e., the guess for $\boldsymbol{x}^{\star}_{1-b,i}-\boldsymbol{x}^{\star}_{b,i}$). These modified encryption keys $\{\mathsf{ek}'_i=\mathbf{s}'_i\}_{i=1}^{\ell}$ are used to answer all encryption queries, to generate the challenge ciphertext and to answer functional decryption key queries as well. At each corruption query i, the adversary is also given ek'_i instead of ek_i.

We first claim that, conditionally on Guess, Game_5''' is statistically close to Game_4''. To see this, we first argue that trading $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ for $\{\mathsf{ek}'_i\}_{i=1}^{\ell}$ has no incidence on queries made by a legitimate adversary:

- We have $\mathbf{C}\cdot\mathbf{s}'_i = \mathbf{C}\cdot\mathbf{s}_i \bmod q$, so that encryption queries obtain the same responses no matter which key set is used among $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ and $\{\mathsf{ek}'_i\}_{i=1}^{\ell}$.
- We have $\sum_{i=1}^{\ell}\mathbf{s}'_i\cdot\boldsymbol{y}_i = \sum_{i=1}^{\ell}\mathbf{s}_i\cdot\boldsymbol{y}_i$ so long as the adversary only obtains private keys for vectors $\boldsymbol{y}\in\mathbb{Z}^{\ell}$ such that $(\mathbf{X}_0^{\star}-\mathbf{X}_1^{\star})\cdot\boldsymbol{y}=\mathbf{0}$ (over $\mathbb{Z}$). Thus functional decryption queries are answered identically, regardless of which key set is used.

- For any corrupted user $i \in \mathscr{CS}$, it should be the case that $\boldsymbol{x}^\star_{0,i} = \boldsymbol{x}^\star_{1,i}$, meaning that $\mathbf{s}'_i = \mathbf{s}_i$ as long as Guess occurs.

This implies that Game'''_5 is identical to Game''_4, except that users' secret keys are defined via (4.12) and thus have a slightly different distribution. Lemma 4.5 shows that the statistical distance between the distributions of $\{\mathbf{s}'_i\}_{i=1}^{\ell}$ and $\{\mathbf{s}_i\}_{i=1}^{\ell}$ is at most $2^{-\lambda}\cdot(4X)^{-n_0\ell}$. This implies that Game''_4 and Game'''_5 are statistically close assuming that Guess occurs. When Guess does not occur, both games output a random $b' \hookleftarrow U(\{0,1\})$, so that $|\Pr[W'''_5] - \Pr[W''_4]| \leq 2^{-\lambda}\cdot(4X)^{-n_0\ell}$.

We finally claim that, from the adversary's view Game'''_5 is identical to Game''_5. Indeed, our choice of $\mathbf{T}$ ensures that $\mathbf{V}\cdot\mathbf{T} = \gamma\cdot\mathbf{I}_{n_0} \bmod q$, so that we have $\mathbf{V}\cdot\mathbf{s}'_i = \mathbf{V}\cdot\mathbf{s}_i + \gamma\cdot(\boldsymbol{x}^\star_{1-b,i} - \boldsymbol{x}^\star_{b,i}) \bmod q$. This implies

$$\begin{aligned}\mathbf{C}_{t^\star,i} &= \mathbf{G}_0^\top\cdot\boldsymbol{x}^\star_{b,i} + (\mathbf{R}_{\tau^\star}^\top\cdot\hat{\mathbf{A}}^\top\cdot\mathbf{C} + \mathbf{G}_0^\top\cdot\mathbf{V})\cdot\mathbf{s}'_i + \mathbf{e}_i{}^\star \\ &= \mathbf{G}_0^\top\cdot\left((1-\gamma)\cdot\boldsymbol{x}^\star_{b,i} + \gamma\cdot\boldsymbol{x}^\star_{1-b,i}\right) + (\mathbf{R}_{\tau^\star}^\top\cdot\hat{\mathbf{A}}^\top\cdot\mathbf{C} + \mathbf{G}_0^\top\cdot\mathbf{V})\cdot\mathbf{s}_i + \mathbf{e}_i{}^\star\end{aligned}$$

which is exactly the distribution from Game''_5.

Putting the above altogether, we find $|\Pr[W''_4] - \Pr[W''_5]| \leq 2^{-\Omega(\lambda)}\cdot(4X)^{-n_0\ell}$, which in turn implies $|\Pr[W_4] - \Pr[W_5]| \leq 2^{-\Omega(\lambda)}$, as claimed.

□

Lemma 4.5. *If $\sigma \geq 2^\lambda\cdot n_0\cdot(4X)^{n_0\ell+1}\cdot\omega(n^2\sqrt{\log n})$, then we have the inequality*

$$\Delta\left(D_{\mathbb{Z}^n,\sigma}, \mathbf{T}\cdot\Delta\boldsymbol{x_i} + D_{\mathbb{Z}^n,\sigma}\right) \leq 2^{-\lambda}\cdot(4X)^{-n_0\ell}.$$

Proof. By Lemma 2.8, Remark 1, each column of $\mathbf{T}$ has norm smaller than:

$$\|\mathbf{t}_i\| \leq O(\sqrt{n\cdot(n_0+n_1)\log q})\cdot\omega(\sqrt{\log n}) = n\cdot\omega\left(\sqrt{\log n}\right),$$

so that $\|\mathbf{T}\cdot\Delta\boldsymbol{x}_i\|_\infty \leq \|\mathbf{T}\cdot\Delta\boldsymbol{x}_i\| \leq 2Xn_0\cdot n\cdot\omega(\sqrt{\log n})$. By Lemma 2.9, the considered distance is smaller than $n\cdot\frac{2Xn_0\cdot n\cdot\omega(\sqrt{\log n})}{\sigma} \leq 2^{-\lambda}\cdot(4X)^{-n_0\ell}$. □

4.4 One-or-less Compiler

4.4.1 Adaptive Multi-Instance PRFs

In order to achieve 1-or-less-IND security (Definition 4.4), our compiler uses as a building block a pseudo-random function family that satisfies a definition of *adaptive multi-instance PRF*. This notion is defined as follows.

Definition 4.5 (ad-mi-PRF)**.** *An efficiently computable function $F : \mathscr{K}\times\mathscr{X}\to\mathscr{Y}$ is an adaptively secure N-instance PRF, for some $N \in \mathsf{poly}(\lambda)$, if no PPT adversary $\mathscr{A}$ has non-negligible advantage in winning the following game.*

Initialization: *The challenger $\mathscr{C}$ samples N secret keys $k_1, k_2, \ldots k_N$ from the key space $\mathscr{K}$ and a uniformly random bit $b \leftarrow U(\{0,1\})$.*

Queries. *The adversary $\mathscr{A}$ adaptively interleaves the following kinds queries:*

- **Evaluation:** *Upon receiving a query* $\mathsf{Eval}(i, X)$*, where $i \in [N]$ and $X \in \mathscr{X}$, the challenger returns $\perp$ if it previously replied to a challenge query for the same pair (i, X). Otherwise, it replies with $F_{k_i}(X) \in \mathscr{Y}$.*
- **Corruption:** *When the adversary makes a query* $\mathsf{Corrupt}(i)$*, the challenger returns $\perp$ if it previously replied to a challenge query for i. Otherwise, the challenger returns $k_i \in \mathscr{K}$.*
- **Challenge:** *$\mathscr{A}$ makes challenge queries* $\mathsf{Challenge}(i, X^\star)$ *for a unique arbitrary input $X^\star$ (any subsequent challenge query involving a different input $X \neq X^\star$ is ignored). If* $\mathsf{Eval}(i, X^\star)$ *or* $\mathsf{Corrupt}(i)$ *was queried before, the challenger returns $\perp$. Else, it returns $F_{k_i}(X^\star)$ if $b = 1$ and a uniformly random value from $\mathscr{Y}$ if $b = 0$.*

Guess. *$\mathscr{A}$ outputs a guess $b' \in \{0,1\}$ and wins if $b' = b$.*

The adversary's advantage is $\mathbf{Adv}^{\textit{ad-mi-PRF}}_{\mathscr{A}}(\lambda) := \left|\Pr[b' = b] - \frac{1}{2}\right|$.

It is possible to show that any PRF that provides single-instance security in the sense of a "find-then-guess" (Definition 2.12) is also secure in the sense of the above Definition 4.5. In short, the proof of Proposition 4.3 proceeds using a hybrid argument over the challenge queries and guesses the index of an uncorrupted index in each hybrid. However, this reduction incurs a quadratic security loss in N.

Proposition 4.3. *If $F : \mathscr{K} \times \mathscr{X} \to \mathscr{Y}$ is a secure PRF (Def. 2.12) then it is also a secure N-instance ad-mi-PRF. Moreover, if there exists an adversary $\mathscr{A}$ that breaks the ad-mi-PRF we give a reduction $\mathscr{B}$ that breaks the standard PRF security such that:*

$$\mathbf{Adv}^{\textit{ad-mi-PRF}}_{\mathscr{A}}(\lambda) \leq N^2 \cdot \mathbf{Adv}^{FG}_{\mathscr{B}}(\lambda)$$

Proof. To prove the above result we use a sequence of hybrid games. We define each game H_k for $k \in \{0, 1, 2, \ldots, N\}$ very similar to the ad-mi-PRF game. In game H_k the challenger starts by sampling N secret keys for the PRF $k_1, k_2, \ldots, k_N$. To answer evaluation and corruption queries that are made in an adaptive manner, the challenger proceeds exactly as in the ad-mi-PRF game. Concretely, for an evaluation query for the pair $(i, X) \in [N] \times \mathscr{X}$, the challenger returns $F(k_i, X)$. Corruption queries for any index i are answered with k_i. Challenge queries are answered differently from the original security game. In game H_k, the first k challenge queries are answered with honest evaluations of the PRF and the rest are answered with uniformly random values. As in the ad-mi-PRF game, corruption queries and challenge queries on the same index are not compatible. Also, the challenge input $X^\star$ should not appear as an evaluation queries.

Notice that an efficient adversary that is able to win the ad-mi-PRF security game is able to distinguish between games H_0 and H_N. We prove below that any two consecutive games are computationally indistinguishable, thus completing the proof. Assume there exits an efficient algorithm $\mathscr{A}_k$ that distinguishes between Game H_k and H_{k+1} with non-negligible advantage. We give an efficient adversary $\mathscr{B}_k$ that breaks the find-then-guess PRF security game.

The reduction $\mathscr{B}_k$ starts by sampling the uniformly random value $i_0 \hookleftarrow U([N])$, which represents a guess for the index for the $(k+1)$-th challenge query that $\mathscr{A}_k$ will make. Then it samples the keys $k_1, k_2, \ldots, k_{i_0-1}, k_{i_0+1}, \ldots, k_N$. Notice the reduction is able to answer corruption, challenge and evaluation queries for any $[N] \setminus \{i_0\}$. If the adversary makes an evaluation query for $(i_0, X) \in [N] \times \mathscr{X}$, the reduction makes an evaluation query for $X \in \mathscr{X}$ to its challenger and receives $F(k_{i_0}, X)$. The value is relayed to the adversary as a response. If a corruption query is made for i_0 the reduction aborts. When the adversary makes a challenge query for $(i_0, X) \in [N] \times \mathscr{X}$, the reduction makes a challenge query on $X^\star$ to its challenger to receive y_b, which is either the honest evaluation $F(k_{i_0}, X^\star)$ or a uniformly random value. The value y_b is given to $\mathscr{A}_k$ as a response to the challenge query. At the end of the game, the reduction outputs the same bit as the adversary $\mathscr{A}_k$. Notice that depending on the nature of the value y_b, the view of the adversary is either as in H_k or H_{k+1}. This means that the advantage of $\mathscr{B}_k$ in breaking the PRF is the same as the advantage of $\mathscr{A}_k$, provided that the guess of i_0 was right. Since the probability of a right guess is $1/N$, and we have N transitions between the hybrid games, we obtain the quadratic loss in security of N^2. □

The N^2 security loss from the above reduction translates into an ℓ^5 loss factor for the actual compiler, where ℓ is the number of clients in the MCFE scheme. This is because Lemma 4.8 uses a reduction with $N = \ell^2$ PRF instances, which gives a degradation factor ℓ^4, which eventually becomes ℓ^5 because of the bound (4.13).

The PRF described in Section 3.4.2 can be proven secure in the sense of Definition 4.5 without the $O(N^2)$ loss in the reduction. This gives a compiler that works under the LWE assumption, with a loss factor that is linear in ℓ, the number of clients. Since our MCFE scheme from Section 4.3.2 relies on the LWE assumption anyway, we do not need an extra assumption in order to use the compiler. In comparison, the compiler of [ABG19] loses a factor $O(\ell^2)$ but works with any PRF.

Theorem 4.6. *The PRF from Section 3.4.2 is also ad-mi-PRF secure, under the* LWE *assumption. Moreover, for any efficient adversary $\mathscr{A}$, there exists an efficient reduction $\mathscr{B}$ such that:*

$$\mathbf{Adv}^{\textit{ad-mi-PRF}}_{\mathscr{A}}(\lambda) \leq n \cdot \mathbf{Adv}^{\mathsf{LWE}_{q,m,n',\alpha}}_{\mathscr{B}}(\lambda) + 2^{-\Omega(\lambda)}.$$

The proof is essentially identical to the proof of Theorem 3.3. Intuitively, it exploits the fact that the reduction knows the secret keys at any time, which makes it easy to consistently answer adaptive corruption queries. The details can be found in [LT19].

4.4.2 The Compiler

The compiler is similar to the random-oracle-based transformation of Abdalla *et al.* [ABKW19, Section 4.2] – which is itself inspired by [ACF+18] – and its intuition is the following. The ciphertexts of individual slots contain partial MCFE ciphertexts $\mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, t)$ which are super-encrypted using a symmetric encryption scheme with secret key $K_{t,i}$. The partial ciphertexts of the compiled scheme also contain shares k_{ijt} of the secret key $K_{i,t}$ in such a way that, once all the partial ciphertexts are gathered for a given tag t, the decryptor can recover the secret $K_{t,i} = \bigoplus_{j=1}^{\ell} k_{ijt}$ and thus the encryption $\mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, i)$. If only one such ciphertext is missing, then the secret key $K_{t,i}$ remains computationally hidden because the decryptor does not have all the shares. The symmetric encryption layer thus hides all the information that could leak when the adversary does not make encryption queries for all uncorrupted slots.

Given a scheme $\mathscr{MCFE}' = (\mathsf{Setup}', \mathsf{Encrypt}', \mathsf{DKeygen}', \mathsf{Decrypt}')$, an adaptively secure ℓ^2-instances ad-mi-PRF $F : \mathscr{K} \times \mathscr{T} \to \{0,1\}^{\lambda}$ and a pseudo-random generator $G : \{0,1\}^{\lambda} \to \{0,1\}^{|\mathsf{ct}'|}$, where $|\mathsf{ct}'|$ is the length of the output of the algorithm $\mathsf{Encrypt}'$, we obtain the following compiled scheme $\mathscr{MCFE}$.

Setup$(\mathsf{cp}, 1^{\ell})$: Runs $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}'(\mathsf{cp}, 1^{\lambda})$. Samples ℓ^2 secret keys $k_{ij} \leftarrow\!\!\$\ \mathscr{K}$ for all $i, j \in [\ell]$. Then, set $\mathsf{mpk} := \mathsf{mpk}'$ $\mathsf{msk} := \mathsf{msk}'$ and $\mathsf{ek}_i := \left(\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j\in[\ell]}\right)$

Encrypt(ek_i, x_i, t) : Given $\mathsf{ek}_i = \left(\mathsf{ek}'_i, \{k_{ij}, k'_{ji}\}_{j=1}^{\ell}\right)$ it computes $k_{ijt} := F_{k_{ij}}(t)$, $k_{jit} := F_{k_{ji}}(t)$ for all $j \in [\ell]$, $C'_{t,i} := \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, t)$ and $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$. Next, it computes $C_{t,i} := C'_{t,i} \oplus G(K_{t,i})$. It outputs $\mathsf{ct}_{t,i} = \left(C_{t,i}, \{k_{jit}\}_{j=1}^{\ell}\right)$.

DKeygen(msk, f) : Given msk and an ℓ-argument function $f : \mathscr{M}^{\ell} \to \mathscr{R}$. It outputs a functional decryption key $\mathsf{dk}_f := \mathsf{DKeygen}'(\mathsf{msk}', f)$.

Decrypt$(\mathsf{dk}_f, t, \{\mathsf{ct}_{t,1}, \mathsf{ct}_{t,2}, \ldots, \mathsf{ct}_{t,\ell}\})$: Takes as input a functional decryption key dk_f, a tag $t \in \mathscr{T}$, and an ℓ-vector of ciphertexts $\mathbf{C} = (\mathsf{ct}_{t,1}, \ldots, \mathsf{ct}_{t,\ell})$. For all $i \in [\ell]$ it computes $K_{t,i} = \bigoplus_{j=1}^{\ell} k_{ijt}$ and $C'_{t,i} := C_{t,i} \oplus G(K_{t,i})$ and runs $\mathsf{Decrypt}'\left(\mathsf{dk}_f, t, \{C'_{t,1}, C'_{t,2}, \ldots, C'_{t,\ell}\}\right)$

The security proof relies on the idea that, if the adversary $\mathscr{A}$ does not make all the allowed challenge encryptions queries, there must exist an honest slot $j_0 \in \mathscr{HS}$ such that $\mathsf{CQEncrypt}(j_0, x^{0\star}_{j_0}, x^{1\star}_{j_0}, t^{\star})$ is never asked. This ensures that none of the PRF values $\{k_{ij_0t^{\star}} = F_{k_{ij_0}}(t^{\star})\}_{i\in\mathscr{HS}}$ is ever exposed to the adversary, which implies that $\{k_{ij_0t^{\star}}\}_{i\in\mathscr{HS}}$ are pseudo-random in the adversary's view since F is a PRF. This implies that secret keys $K_{t^{\star},i} := \bigoplus_{j=1}^{\ell} k_{ijt^{\star}}$ are also pseudo-random for all $i \in \mathscr{HS}$. Hence, the ciphertexts $C_{t^{\star},i} = \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, i) \oplus G(K_{t^{\star},i})$ computationally hide the x_i's for all uncorrupted slots i. To translate this intuition into a

formal security proof, we need a PRF family that provides security in the sense of Definition 4.5.

Theorem 4.7. *If $\mathcal{MCFE}'$ is 1ch-IND secure then the compiled $\mathcal{MCFE}$ scheme is 1-or-less-IND secure. Moreover, if $\mathcal{A}$ breaks the security of the $\mathcal{MCFE}$ scheme, we give a reduction $\mathcal{B}$ that breaks the security of the $\mathcal{MCFE}'$ such that:*

$$\mathbf{Adv}^{1\text{-}or\text{-}less}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{1ch\text{-}IND}_{\mathcal{B}}(\lambda) + \ell \cdot \mathbf{Adv}^{ad\text{-}mi\text{-}PRF}(\lambda) + \mathbf{Adv}^{PRG}(\lambda)$$

Proof. Given an adversary $\mathcal{A}$ that breaks the $\mathcal{MCFE}$ scheme by winning the game of Definition 4.4 with non-negligible probability, we give an efficient reduction $\mathcal{B}$ that wins the game of Definition 4.3 with noticeable advantage, thus breaking the $\mathcal{MCFE}'$ scheme. The algorithm $\mathcal{B}$ works as follows.

Initialization: The challenger $\mathcal{C}$ gives $\mathcal{B}$ the public parameters mpk' and keeps to itself the secret keys $(\mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell})$. In order to initialize the game with $\mathcal{A}$, the reduction $\mathcal{B}$ samples ℓ^2 secret keys $k_{ij} \leftarrow\!\!\$\ \mathcal{K}$ for all $i, j \in [\ell]$. Then gives $\mathcal{A}$ the public parameters $\mathsf{mpk} := \mathsf{mpk}'$.

Encryption queries: For each encryption query $\mathsf{QEncrypt}(i, x, t)$ made by $\mathcal{A}$, $\mathcal{B}$ sends to same query to its challenger and obtains $C'_{t,i} = \mathsf{Encrypt}'(\mathsf{ek}'_i, x, t)$. Then, $\mathcal{B}$ computes $\left\{k_{ijt} := F_{k_{ij}}(t), k_{jit} := F_{k_{ji}}(t)\right\}_{j=1}^{\ell}$, $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$, $C_{t,i} := C'_{t,i} \oplus G(K_{t,i})$ and gives $\mathsf{ct}_{t,i} := (C_{t,i}, \{k_{jit}\}_{j=1}^{\ell})$ to $\mathcal{A}$.

Challenge queries: If $\mathcal{A}$ makes a query $\mathsf{CQEncrypt}(i, x_i^{\star 0}, x_i^{\star 1}, t^\star)$, $\mathcal{B}$ relays it to its challenger and obtains $C'_{t^\star,i} := \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i^{\star b}, t^\star)$. Then, $\mathcal{B}$ computes $\left\{k_{ijt^\star} := F_{k_{ij}}(t^\star), k_{jit^\star} := F_{k_{ji}}(t^\star)\right\}_{j=1}^{\ell}$, $K_{t^\star i} := \bigoplus_{j=1}^{\ell} k_{ijt^\star}$, as well as $C_{t^\star,i} := C'_{t^\star,i} \oplus G(K_{t^\star i})$. The adversary $\mathcal{A}$ is given $\mathsf{ct}_{t^\star i} := (C_{t^\star i}, \{k_{jit^\star}\}_{j=1}^{\ell})$.

Functional decryption key queries: Each query $\mathsf{QDKeygen}(f)$ made by $\mathcal{A}$ is relayed by $\mathcal{B}$ to its challenger, which replies with dk'_f. Then, $\mathcal{B}$ gives $\mathsf{dk}_f := \mathsf{dk}'_f$ to $\mathcal{A}$.

Corruption queries: For each corruption query $\mathsf{QCorrupt}(i)$ that $\mathcal{A}$ makes, the reduction $\mathcal{B}$ replies with $\mathsf{ek}_i := \left(\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j=1}^{\ell}\right)$, where ek'_i is the key given by its challenger in response to the same corruption query.

Finalize: When $\mathcal{A}$ outputs a result b', $\mathcal{B}$ distinguishes two situations.

- If $\mathcal{A}$ makes queries $\mathsf{CQEnc}(i, \star, \star, t^\star)$ for all slots $i \in \mathcal{HS}$ and the conditions of the Finalize step in Definition 4.4 are all satisfied (i.e., $\mathcal{A}$'s output is not overwritten by a random bit β), $\mathcal{B}$ outputs the same bit $\tilde{b} := b'$ as $\mathcal{A}$.
- Otherwise, $\mathcal{B}$ outputs a random $\tilde{b} \leftarrow\!\!\$\ U(\{0,1\})$. We call this event Abort.

Next, we analyze the probability of $\mathscr{B}$ winning the game.

$$\begin{aligned}
\Pr[\tilde{b} = b] &= \Pr[\tilde{b} = b | \mathsf{Abort}] \cdot \Pr[\mathsf{Abort}] + \Pr[\tilde{b} = b | \overline{\mathsf{Abort}}] \cdot \Pr[\overline{\mathsf{Abort}}] \\
&= \frac{1}{2} \cdot \Pr[\mathsf{Abort}] + \Pr[b' = b | \overline{\mathsf{Abort}}] \cdot \Pr[\overline{\mathsf{Abort}}] \\
&= \Pr[b' = b] + \left(\frac{1}{2} - \Pr[b' = b | \mathsf{Abort}] \right) \cdot \Pr[\mathsf{Abort}] \\
&= \Pr[b' = b] + \left(\frac{1}{2} - \Pr[b' = b | \mathsf{abort}] \right) \cdot \Pr[\mathsf{abort}]
\end{aligned}$$

where abort is the event that $\mathscr{B}$ aborts the attack because $\mathscr{A}$ did not make all the challenge queries for all honest slots (but all the other conditions from the Finalize step in Def. 4.4 are satisfied). We can replace the event Abort with the event abort in the equality above because, in the case $\mathscr{A}$ does not meet at least one of the other conditions from the Finalize step, its output is the same as $\mathscr{B}$'s, namely a uniformly random bit.

In order to finish the proof, we need to prove the following claim.

Claim. *If F is an ad-mi-PRF and G a PRG, then $\left| \Pr[b' = b | \mathsf{abort}] - \frac{1}{2} \right|$ is negligible.*

Proof. Since we are in the setting where abort happens, we can consider that the class of adversaries $\mathscr{A}$ is restricted to those who always satisfy the conditions of the event abort, namely it does not make all the encryption queries for all honest slots, but meets all the other conditions in the Finalize step. We will prove this claim via a series of games. For each $i \in \{0,1,2\}$, we call W_i the event that $b' = b$ in Game_i, i.e. the adversary $\mathscr{A}$ makes the correct guess and wins the game.

Game_0: This is the original game in the one-or-less security experiment (Definition 4.4) for the compiled scheme. The challenger uniformly samples a bit $b \leftarrow\!\!\$\ U(\{0,1\})$ and then runs the setup algorithm of $\mathscr{MCFE}'$ to obtain $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell})$, samples $k_{ij} \leftarrow\!\!\$\ \mathscr{K}$, for all $i, j \in [\ell]$ and sets $\mathsf{mpk} := \mathsf{mpk}'$, $\mathsf{msk} := \mathsf{msk}'$ and $\mathsf{ek}_i := \left(\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j=1}^{\ell}\right)$. It gives mpk to $\mathscr{A}$ and handles queries $\mathsf{QCorrupt}(i)$ using $\mathsf{ek}_i := (\mathsf{ek}'_i, \{k_{ij}, k_{ji}\}_{j=1}^{\ell})$. To answer queries $\mathsf{QEnc}(i, x_i, t)$, it computes the shares $\left\{k_{ijt} := F_{k_{ij}}(t), k_{jit} := F_{k_{ji}}(t)\right\}_{j=1}^{\ell}$ and the seed $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$. Then, it computes $C_{t,i} := \mathsf{Encrypt}'(\mathsf{ek}'_i, x_i, t) \oplus G(K_{t,i})$. The adversary $\mathscr{A}$ is given $\mathsf{ct}_{t,i} := \left(C_{t,i}, \{k_{jit}\}_{j=1}^{\ell}\right)$. To answer a challenge encryption query $\mathsf{CQEncrypt}(i, x_i^0, x_i^1, t^\star)$, it replies with $\mathsf{QEnc}(i, x_i^b, t^\star)$. For functional decryption queries $\mathsf{QDKeygen}(f, \mathsf{msk})$, the challenger replies with $\mathsf{dk}_f \leftarrow \mathsf{DKeygen}'(\mathsf{msk}', f)$. When $\mathscr{A}$ halts, it outputs a bit b', so $\Pr[W_0] = \Pr[b' = b | \mathsf{abort}]$ by definition.

Game_1: The challenger interacts with $\mathscr{A}$ exactly as in Game_0 except that, upon receiving a challenge query $\mathsf{CQEncrypt}(i, x_i^0, x_i^1, t^\star)$, it does the following. If

$x_i^0 = x_i^1$, it replies as in Game_0. Otherwise, for each slot $i \in [\ell]$ such that $x_i^0 \neq x_i^1$, it samples a uniform value for $K_{t^\star,i} \leftarrow U(\{0,1\}^\lambda)$ which it uses as a seed for the pseudo-random generator G in order to compute $C_{t^\star,i}$.

Since we are conditioning on the event abort, there exists an index $j_0 \in \mathcal{HS}$ which is never the input to a query of the form $\mathsf{CQEnc}(j_0,\cdot,\cdot,t^\star)$. Moreover, for any index i such that $x_i^0 \neq x_i^1$, corruption queries $\mathsf{Corrupt}(i)$ are disallowed by the conditions of the Finalize step which must be satisfied (recall that we are conditioning on event abort). The only information that $\mathcal{A}$ can gather about $K_{t^\star,i}$ is thus obtained from $\mathsf{QEnc}(j,x_j,t)$ and $\mathsf{CQEnc}(j,x_j^0,x_j^1,t^\star)$ queries. By making these types of queries, $\mathcal{A}$ learns any $\{k_{ijt} = F_{k_{ij}}(t)\}$ for any $i,j \in [\ell]$, any $t \in \mathcal{T} \setminus \{t^\star\}$ and $\left\{F_{k_{ij}}(t^\star)\right\}_{j \in \mathcal{HS} \setminus \{j_0\}}$, respectively. Intuitively the pseudo-randomness of F should be enough to prove that $F_{k_{ij_0}}(t^\star)$ is pseudo-random which implies that the key $K_{t^\star i}$ is also pseudo-random, and this assures that Game_0 and Game_1 are indistinguishable.

In order to prove that Game_0 and Game_1 are indeed computationally indistinguishable, we make use of another two intermediate games, Game_0' and Game_1' that are proved to be computationally indistinguishable. They are defined below.

Game_d' ($d \in \{0,1\}$): This game is identical to Game_d with the difference that the challenger initially chooses $j_0 \leftarrow U([\ell])$ as a guess for the smallest index $j_0 \in \mathcal{HS}$ such that no challenge query $\mathsf{CQEnc}(j_0,\cdot,\cdot,t^\star)$ is ever made. As soon as the guess turns out to be wrong, the challenger stops the interaction and outputs a uniformly random bit $b' \leftarrow U(\{0,1\})$, instead of an output provided by $\mathcal{A}$. For each $d \in \{0,1\}$, we have $\Pr[W_d'] = (\ell-1)/2\ell + 1/\ell \cdot \Pr[W_d]$, which is equivalent to:

$$\Pr[W_d] = \ell \cdot \left(\Pr[W_d'] - \frac{1}{2}\right) + \frac{1}{2}$$

Notice that this implies $|\Pr[W_0] - \Pr[W_1]| = \ell \cdot |\Pr[W_0'] - \Pr[W_1']|$, where the event W_d' is the event that $b' = b$ in Game_d'.

Lemma 4.8 shows that Game_0' and Game_1' are computationally indistinguishable if F is a secure PRF in the multi-instance setting and G is a secure pseudo-random generator.

Game_2: This game is identical to Game_1 except that the challenger responds to challenge queries of the form $\mathsf{CQEncrypt}(i,x_i^0,x_i^1,t^\star)$ such that $x_i^0 \neq x_i^1$, with a uniformly random $C_{t^\star,i}$, as part of the ciphertext. In this game, we obviously have $\Pr[W_2] = \Pr[b' = b|\mathsf{abort}] = \frac{1}{2}$, because the cipheretext does not contain any information about the bit b.

To see that Game_1 and Game_2 are computationally indistinguishable, recall that $C_{t^\star,i} = C'_{t^\star,i} \oplus G(K_{t^\star,i})$. When the seed $K_{t^\star,i}$ is uniformly generated, $G(K_{t^\star,i})$ is pseudo-random, hence Game_1 and Game_2 are computationally indistinguishable. In particular, we thus obtain $|\Pr[W_2] - \Pr[W_1]| \leq \mathbf{Adv}^{\mathrm{PRG}}(\lambda)$.

When combining the above, we obtain

$$|\Pr[W_0] - \Pr[W_2]| \leq \ell \cdot \mathbf{Adv}^{\text{ad-mi-PRF}}(\lambda) + \mathbf{Adv}^{\mathrm{PRG}}(\lambda), \tag{4.13}$$

which proves the result. □

Lemma 4.8. *If F is secure in the* ad-mi-PRF *sense, then* Game'_0 *and* Game'_1 *are computationally indistinguishable. More precisely, for any distinguisher $\mathscr{A}$, there exists a PRF adversary $\mathscr{B}$ such that* $\mathbf{Adv}_{\mathscr{A}}(\lambda) \leq \mathbf{Adv}^{\text{ad-mi-PRF}}_{\mathscr{B}}(\lambda)$.

Proof. Suppose that $\mathscr{A}$ is adistinguisher between the two games. We construct an adversary $\mathscr{B}$ that breaks the ad-mi-PRF security of F when the challenger samples $N = \ell^2$ secret keys $k_{ij} \leftarrow\!\!\$\ \mathcal{K}$ for all $i, j \in [\ell]$ in the game of Definition 4.5. Algorithm $\mathscr{B}$ proceeds in the following way.

Initialization: $\mathscr{B}$ starts by drawing $j_0 \leftarrow\!\!\$\ U([\ell])$ as a guess that j_0 is the smallest index such that no challenge query $\mathsf{CQEncrypt}(j_0, \cdot, \cdot, t^\star)$ is made. Then, it runs $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i=1}^{\ell}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{mpk} := \mathsf{mpk}'$ is given to $\mathscr{A}$.

Corruption queries: In order to answer a corruption query $\mathsf{Corrupt}(j)$, if $j = j_0$ the reduction $\mathscr{B}$ stops the attack, since its guess for j_0 was incorrect and outputs the same bit as the distinguisher in this situation. Otherwise, it obtains the values $\{k_{j\tau}, k_{\tau j}\}_{\tau=1}^{\ell}$ by making corruption queries to its PRF challenger. Then, it replies with $(\mathsf{ek}'_j, \{k_{j\tau}, k_{\tau j}\}_{\tau=1}^{\ell})$. Notice that, if j_0 is a correct guess, we have $j_0 \in \mathcal{HS}$, which implies that the adversary never learns $\{k_{j_0 i}, k_{i j_0}\}_{i \in \mathcal{HS}}$ from corruption queries.

Encryption queries: To answer $\mathsf{QEncrypt}(i, x, t)$ queries, $\mathscr{B}$ asks its PRF challenger for the evaluations $\left\{k_{ijt} := F_{k_{ij}}(t), k_{jit} := F_{k_{ji}}(t)\right\}_{j=1}^{\ell}$ and computes $K_{t,i} := \bigoplus_{j=1}^{\ell} k_{ijt}$. Then, it computes $C'_{t,i} = \mathsf{Encrypt}'(\mathsf{ek}'_i, x, t)$ as well as $C_{t,i} := C'_{t,i} \oplus G(K_{t,i})$. The ciphertext $\mathsf{ct}_{t,i} := (C_{t,i}, \{k_{jit}\}_{j=1}^{\ell})$ is given to $\mathscr{A}$.

Challenge queries: When receiving a challenge query $\mathsf{CQEncrypt}(i, x_i^{0\star}, x_i^{1\star}, t^\star)$, $\mathscr{B}$ stops the interaction if $i = j_0$ and outputs the same guess bit b' as the distinguisher would in this situation. Otherwise, it considers the following two cases.

- If $x_i^{0\star} = x_i^{1\star} = x_i^\star$, then $\mathscr{B}$ responds as it would answer the encryption query $\mathsf{QEncrypt}(i, x_i^\star, t^\star)$.

- If $x_i^{0\star} \neq x_i^{1\star}$ (which is only possible if $i \in \mathcal{HS}$), the reduction $\mathcal{B}$ asks its challenger for the PRF evaluations $\{k_{ijt^\star} := F_{k_{ij}}(t^\star)\}_{j\neq j_0}$. It also makes a challenge query for the input $t^\star$ evaluated with the secret key k_{ij_0}. The challenger replies with a value $y_{ij_0t^\star}$, which is either the evaluation $F_{k_{ij_0}}(t^\star)$ for the unknown secret key k_{ij_0} chosen by the PRF challenger or a random value. Then, $\mathcal{B}$ computes $K_{t^\star,i} := (\bigoplus_{j\neq j_0} k_{ijt^\star}) \oplus y_{ij_0t^\star}$ and uses $K_{t^\star,i}$ as a seed for the PRG, by computing $C^b_{t^\star,i} := C^{b\prime}_{t^\star,i} \oplus G(K_{t^\star,i})$. The adversary $\mathcal{A}$ is given $\mathsf{ct}_{t,i} := \left(C^b_{t,i}, \{k_{jit^\star}\}_{j=1}^{\ell}\right)$.

Guess: The reduction $\mathcal{B}$ always outputs the same guess bit b' as as the distinguisher $\mathcal{A}$, when the interaction stops.

We now observe that, if $\mathcal{B}$'s challenger always returns truly random values $y_{ij_0t^\star}$ at each challenge query, then $\mathcal{A}$'s view is identical to that of Game'_1. If $\mathcal{B}$'s challenger always returns pseudo-random values, $\mathcal{A}$'s view is as in Game'_0. We thus conclude that $\mathbf{Adv}^{\text{ad-mi-PRF}}_{\mathcal{B}}(\lambda) \geq \mathbf{Adv}_A(\lambda)$. □

□

Chapter

5

Adaptive Simulation Security of Inner Product Functional Encryption

In this chapter we present the work from [ALMT20] which shows that the DDH-based scheme of [ALS16] and some variants of the DCR-based and the LWE-based Inner Product Functional Encryption schemes of [ALS16] can actually be proved secure in the *adaptive-simulation* security model, for an unbounded number of key queries.

Since the impossibility result of [BSW11a] can be adapted to the inner-product functionality, it excludes the possibility of achieving AD-SIM security when the adversary is allowed to issue multiple challenge messages. However, none of the impossibility results from [BSW11a, O'N10, AGVW13] applies to our case. Therefore, we prove that the above-mentioned DDH, DCR and LWE-based IPFE schemes achieve the strongest security notion that we can hope for among the IND and SIM based definitions, namely it provides AD-SIM security for single challenge ciphertexts and unbounded number of key queries. Simulation-based security has been proven before for the DDH-based scheme of [ALS16], but only in the selective case [AGRW17] or the semi-adaptive case [Wee17].

Our main insight is that the semi-adaptive simulator of [Wee17], that worked only for the DDH-based scheme of [ALS16], can be modified to answer any pre-challenge key-queries as well, thus, proving AD-SIM security for the same scheme. We show that the same ideas can be used to prove AD-SIM security for the DCR and LWE-based schemes of [ALS16], with some modifications.

Organization

We begin this chapter with Section 5.1, where we give the syntax and the security definitions for IPFE. In Section 5.2 we recall the DDH-based scheme of [ALS16] and prove its AD-SIM security. Next, we show in Section 5.3 that a variant of the DCR-based scheme from the same work, satisfies AD-SIM security. Finally, Section 5.4 deals with similar results for a modified LWE-based scheme of [ALS16].

5.1 Definitions

In this section we recall the syntax of Functional Encryption [BSW11a] as well as both the IND and SIM security security definitions.

5.1.1 Functional Encryption (FE)

Definition 5.1. *A Functional Encryption (FE) scheme over a class of functions* $\mathscr{F} = \{f : \mathscr{X} \to \mathscr{Z}\}$ *consists of the* PPT *algorithms* (Setup, Keygen, Encrypt, Decrypt)*:*

Setup$(1^\lambda, \mathscr{F})$: *Outputs a public key* mpk *and a master secret key* msk.

Keygen(msk, f) : *Given the master secret key and a functionality* $f \in \mathscr{F}$*, the algorithm outpus a secret key* sk_f.

Encrypt$(\mathsf{mpk}, \boldsymbol{x})$: *On input the public key and a message* $\boldsymbol{x} \in \mathscr{X}$ *from the message space, the algorithm outputs a ciphertext* $\boldsymbol{c}$.

Decrypt$(\mathsf{mpk}, \mathsf{sk}_f, \boldsymbol{c})$: *Given a ciphertext and a secret key corresponding to some functionality* $f \in \mathscr{F}$*, the algorithm outputs* $\boldsymbol{z} \in \mathscr{Z}$.

Correctness. For $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathscr{F})$, for any $\boldsymbol{x} \in \mathscr{X}$, any $f \in \mathscr{F}$, $\boldsymbol{c} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x})$ and $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$, we require that $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{sk}_f, \boldsymbol{c}) = f(\boldsymbol{x})$, with overwhelming probability.

5.1.2 Security

Next, we define security of functional encryption. Security comes in two flavors – indistinguishability-based and simulation-based – we define each in turn.

We first define the weaker indistinguishability-based security [BSW11a]. In this notion, one asks that no efficient adversary be able to differentiate encryptions of $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ without obtaining secret keys sk_f such that $f(\boldsymbol{x}_0) \neq f(\boldsymbol{x}_1)$.

Definition 5.2 (AD-IND)**.** *A functional encryption scheme is given by four* PPT *algorithms* $\mathscr{FE}$ = (Setup, Keygen, Encrypt, Decrypt) *provides semantic security under chosen-plaintext attacks (or IND-CPA security) if no* PPT *adversary has non-negligible advantage in the following game:*

1. *The challenger runs* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and the master public key* mpk *is given to the adversary* $\mathscr{A}$.

2. *The adversary adaptively makes secret key queries to the challenger. At each query, adversary* $\mathscr{A}$ *chooses* $f \in \mathscr{F}$ *and obtains the key* $\mathsf{sk}_f \leftarrow \mathsf{Keygen}(\mathsf{msk}, f)$.

3. *Adversary* $\mathscr{A}$ *chooses distinct messages* $\boldsymbol{x}_0, \boldsymbol{x}_1$ *subject to the restriction that* $f(\boldsymbol{x}_0) = f(\boldsymbol{x}_1)$*, for any* f *queried in Stage* 2*. Then, the challenger flips a fair coin* $b \hookleftarrow \{0,1\}$ *and computes* $\boldsymbol{c}^\star \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x}_b)$ *which is sent as a challenge to* $\mathscr{A}$.

4. *Adversary* $\mathscr{A}$ *makes further secret key queries for arbitrary functions* $f \in \mathscr{F}$*. However, it is required that* $f(\boldsymbol{x}_0) = f(\boldsymbol{x}_1)$ *at each query.*

5. *Adversary* $\mathscr{A}$ *eventually outputs a bit* $b' \in \{0,1\}$ *and wins if* $b' = b$.

The adversary's advantage is defined to be $\mathbf{Adv}_{\mathscr{A}}(\lambda) := |\Pr[b' = b] - 1/2|$*, where the probability is taken over all coin tosses.*

Definition 5.2 captures *adaptive* security in that the adversary is allowed to choose the messages $\boldsymbol{x}_0, \boldsymbol{x}_1$ at Stage 3.

As pointed out in [BSW11a], indistinguishability-based security is not fully satisfactory in general as it may fail to rule out constructions that are intuitively insecure. They argue that, whenever it is possible at all, one should prefer a stronger notion of simulation-based security. We recall this notion hereunder.

Definition 5.3 (AD-SIM)**.** *For a* FE *scheme defined as above, a* PPT *adversary* $\mathscr{A} = (A_1, A_2)$ *and a* PPT *simulator* $\mathsf{Sim} = (\mathsf{Setup}^\star, \mathsf{Keygen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{Keygen}_1^\star)$*, we define the following experiments:*

$\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{FE},\mathscr{A}}(1^\lambda)$	$\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{FE},\mathscr{A}}(1^\lambda)$
1. $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathscr{F})$	1. $(\mathsf{mpk}^\star, \mathsf{msk}^\star) \leftarrow \mathsf{Setup}^\star(1^\lambda, \mathscr{F})$
2. $(\boldsymbol{x}^\star, \mathsf{st}) \leftarrow A_1^{\mathsf{Keygen}(\mathsf{msk},\cdot)}(\mathsf{mpk})$	2. $(\boldsymbol{x}^\star, \mathsf{st}) \leftarrow A_1^{\mathsf{Keygen}_0^\star(\mathsf{msk}^\star,\cdot)}(\mathsf{mpk}^\star)$ *Let* $\mathscr{V} = \{(f_i, f_i(\boldsymbol{x}^\star), \mathsf{sk}_{f_i})\}_{i=1}^k$
3. $\boldsymbol{c} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x}^\star)$	3. $(\boldsymbol{c}^\star, \mathsf{st}') \leftarrow \mathsf{Encrypt}^\star(\mathsf{msk}^\star, \mathscr{V}, 1^{\lvert\boldsymbol{x}^\star\rvert})$
4. $\alpha \leftarrow A_2^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \boldsymbol{c}, \mathsf{st})$	4. $\alpha \leftarrow A_2^{\mathsf{KeyGen}_1^\star(\mathsf{msk}^\star,\mathsf{st}',\cdot)}(\mathsf{mpk}^\star, \boldsymbol{c}^\star, \mathsf{st})$
5. *Output* α	5. *Output* α

In the Ideal *experiment above, the* $\{f_i \in \mathscr{F}\}_{i=1}^{k}$ *are the functionalities for which the adversary requests their corresponding keys,* $\{\mathsf{sk}_{f_i}\}_{i=1}^{k}$. *An* FE *scheme achieves adaptive simulation-based (*AD-SIM*) security if there exists a* PPT *simulator* Sim *such that, for any* PPT *adversary* $\mathscr{A}$*, the distributions* $\mathsf{Exp}^{\mathsf{Real}}_{\mathsf{FE},\mathscr{A}}(1^\lambda)$ *and* $\mathsf{Exp}^{\mathsf{Ideal}}_{\mathsf{FE},\mathscr{A}}(1^\lambda)$ *are computationally indistinguishable.*

5.2 Adaptive Simulation Security from DDH

In this section, we first recall the IPFE scheme of [ALS16]. Abdalla *et al.* [AGRW17] previously showed that this construction provides simulation-based security for selective adversaries. In [Wee17], Wee gave a proof of simulation-based security for semi-adaptive adversaries. In the semi-adaptive model adversaries are restricted to make all of the key queries after the challenge. We provide a proof that handles adaptive adversaries without any modification in the original scheme.

5.2.1 Overview

In the proof of [Wee17] the simulator creates the dummy message by encrypting the zero vector. To answer a post-challenge key query (the only type of key query allowed in the *semi-adaptive* model), the simulator has to embed the value $z_{\boldsymbol{y}} = f_{\boldsymbol{y}}(\boldsymbol{x}^\star) = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$, where $\boldsymbol{x}^\star$ is the challenge message, in the simulated key $(s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$ in order to correctly explain decryption. This is done through a linear shift of the keys that are used in the real scheme. Namely if the master secret key, corresponding to the master public key $g^{\boldsymbol{s}} \cdot h^{\boldsymbol{t}}$, is given by $(\boldsymbol{s}, \boldsymbol{t}) \in \mathbb{Z}_q^\ell \times \mathbb{Z}_q^\ell$, then the functional decryption key used in the real scheme is composed of $(\langle \boldsymbol{s}, \boldsymbol{y} \rangle, \langle \boldsymbol{t}, \boldsymbol{y} \rangle)$. The simulated functional secret keys are computed as:

$$s'_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \alpha \cdot \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle \text{ and } t'_{\boldsymbol{y}} := \langle \boldsymbol{t}, \boldsymbol{y} \rangle + \beta \cdot \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle,$$

where $\alpha, \beta \in \mathbb{Z}_q$ are carefully chosen.

In the view of this adversary this is the same as replacing the master secret key with $\boldsymbol{s}' := \boldsymbol{s} + \alpha \cdot \boldsymbol{x}^\star$ and $\boldsymbol{t}' := \boldsymbol{t} + \beta \cdot \boldsymbol{x}^\star$, while keeping invariant the master public key and also keeping the responses to key queries consistent. The proof from [Wee17] shows that the under the DDH assumption the real and the ideal experiment are indistinguishable in the semi-adaptive model. This means that the adversary can adaptively chose the challenge message $\boldsymbol{x}^\star$, after seeing the public parameters, but before making any of the key queries.

To prove security in the stronger AD-SIM model, where the adversary is allowed to choose the challenge after making key queries, we follow the same approach as [Wee17], but in the simulation we need to encrypt a dummy message, without having access to the challenge $\boldsymbol{x}^\star$, that is consistent not only with post-challenge keys, but also with the pre-challenge keys. We do this by answering pre-challenge key queries as in the real scheme and we have the simulator answer the

challenge by encrypting a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^\ell$ (instead of the zero vector as in [Wee17]) that is compatible with the challenge message $\boldsymbol{x}^\star$ when decrypting with any pre-challenge key. Namely, we compute $\bar{\boldsymbol{x}}$ such that $\langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle \bmod q$ for all the pre-challenge key queries $\boldsymbol{y} \in \mathbb{Z}_q^\ell$. This is equivalent to solving the system $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}}$, which can easily be done by doing linear algebra over $\mathbb{Z}_q$.

Once the simulator is committed to the challenge ciphertext, it has to "program" the post-challenge keys such that they decrypt the dummy ciphertext to the real function evaluations $z_{\boldsymbol{y}} = f_{\boldsymbol{y}}(\boldsymbol{x}^\star) = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$. Given a post-challenge query $\boldsymbol{y} \in \mathbb{Z}_q^\ell$ and the corresponding function evaluation $z_{\boldsymbol{y}} = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$, the value $z_{\boldsymbol{y}}$ is embedded in the simulated functional key in such a way that the difference $z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle$ between $z_{\boldsymbol{y}}$ and the function evaluation $f_{\boldsymbol{y}}(\bar{\boldsymbol{x}})$ serves as a shift of the real actual key: namely, the simulator returns $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$, where

$$\begin{aligned} s'_{\boldsymbol{y}} &:= \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle) \bmod q \\ t'_{\boldsymbol{y}} &:= \langle \boldsymbol{t}, \boldsymbol{y} \rangle + \beta \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle) \bmod q. \end{aligned} \tag{5.1}$$

By exploiting the linearity properties of the scheme, the shift terms $\alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle)$ and $\alpha \cdot (z_{\boldsymbol{y}} - \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle)$ ensure that $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$ will decrypt the dummy ciphertext to the oracle-supplied $z_{\boldsymbol{y}}$. As in [Wee17], we can prove that this shift of post-challenge keys is equivalent to a shift of the master secret key from the adversary's view: namely, $\mathsf{msk} = (\boldsymbol{s}, \boldsymbol{t})$ is traded for $\mathsf{msk}' = (\boldsymbol{s}', \boldsymbol{t}')$, where $\boldsymbol{s}' = \boldsymbol{s} + \alpha \cdot (\boldsymbol{x}^\star - \tilde{\boldsymbol{x}})$ and $\boldsymbol{t}' = \boldsymbol{t} + \beta \cdot (\boldsymbol{x}^\star - \tilde{\boldsymbol{x}})$. By applying complexity leveraging to a statistical argument [Wee14, BBL17], we can prove that the two master secret keys of $(\boldsymbol{s}', \boldsymbol{t}')$ and $(\boldsymbol{s}, \boldsymbol{t})$ are identically distributed in the adversary's view, even if $\boldsymbol{x}^\star$ is chosen adaptively after it has seen the public parameters and pre-challenge queries.

5.2.2 The DDH-based Construction

Below we recall the description of the [ALS16] IPFE scheme that works for the class of bounded inner-product functionalities:

$$\mathscr{F} = \left\{ f_{\boldsymbol{y}} : ([-X, X] \cap \mathbb{Z})^\ell \to \mathbb{Z} : \boldsymbol{y} \in ([-Y, Y] \cap \mathbb{Z})^\ell \text{ and } f_{\boldsymbol{y}}(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \right\}.$$

It's worth mentioning that the only reason we need bounded inner-products for this scheme is to be able to solve the discrete logarithm when decrypting.

Setup$(1^\lambda, 1^\ell)$**:** Choose a cyclic group $\mathbb{G}$ of prime order $q > 2^\lambda$ and the generators $g, h \leftarrow\!\!\!\!\!\!\rightarrow U(\mathbb{G})$. Then, for each $i \in \{1, \ldots, \ell\}$, sample $s_i, t_i \leftarrow\!\!\!\!\!\!\rightarrow U(\mathbb{Z}_q)$ and compute $h_i = g^{s_i} \cdot h^{t_i}$. Define $\mathsf{msk} := \{s_i, t_i\}_{i=1}^\ell$ and

$$\mathsf{mpk} := \left(\mathbb{G}, g, h, \{h_i\}_{i=1}^\ell \right).$$

Keygen$(\mathsf{msk}, \boldsymbol{y})$**:** To generate a key for the vector $\boldsymbol{y} = (y_1, \ldots, y_\ell) \in \mathbb{Z}^\ell$ such that $\|\boldsymbol{y}\|_\infty \leq Y$, compute $(s_{\boldsymbol{y}}, t_{\boldsymbol{y}}) = (\sum_{i=1}^\ell s_i \cdot y_i, \sum_{i=1}^\ell t_i \cdot y_i) = (\langle \boldsymbol{s}, \boldsymbol{y} \rangle, \langle \boldsymbol{t}, \boldsymbol{y} \rangle)$ and return $\mathsf{sk}_{\boldsymbol{y}} := (\boldsymbol{y}, s_{\boldsymbol{y}}, t_{\boldsymbol{y}})$.

Encrypt$(\mathsf{mpk}, \boldsymbol{x})$**:** To encrypt a vector $\boldsymbol{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}^\ell$ that satisfies $\|\boldsymbol{x}\|_\infty \leq X$, sample $r \leftarrow \mathbb{Z}_q$ and compute

$$C = g^r, \qquad D = h^r, \qquad \{E_i = g^{x_i} \cdot h_i^r\}_{i=1}^\ell.$$

Return $C_{\boldsymbol{x}} = (C, D, E_1, \ldots, E_\ell)$.

Decrypt$(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, C_{\boldsymbol{x}})$**:** Given $\mathsf{sk}_{\boldsymbol{y}} = (\boldsymbol{y}, s_{\boldsymbol{y}}, t_{\boldsymbol{y}})$, compute

$$E_{\boldsymbol{y}} = (\prod_{i=1}^{\ell} E_i^{y_i}) / (C^{s_{\boldsymbol{y}}} \cdot D^{t_{\boldsymbol{y}}}).$$

Then, compute and output $\log_g(E_{\boldsymbol{y}})$.

Correctness. Note that $\prod_{i=1}^{\ell} E_i^{y_i} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g^{r\langle \boldsymbol{s}, \boldsymbol{y} \rangle} \cdot h^{r\langle \boldsymbol{t}, \boldsymbol{y} \rangle} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot C^{s_{\boldsymbol{y}}} \cdot D^{t_{\boldsymbol{y}}}$, which implies $E_{\boldsymbol{y}} = g^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$. The decryption algorithm can thus recover $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \bmod q$ by solving a discrete logarithm instance in a small interval, by restricting messages and keys so as to have $|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq L$, for some polynomially bounded $L = \ell \cdot XY \in \mathsf{poly}(\lambda)$. In this case, the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ can be recovered in $\tilde{O}(L^{1/2})$ time using [Pol00].

5.2.3 The Security Proof

Theorem 5.1. *The scheme provides simulation-based security against adaptive adversaries under the* DDH *assumption.*

Proof. To prove the result, we first describe a PPT simulator before showing that, under the DDH assumption, the adversary cannot distinguish the ideal experiment from the real experiment.

In both experiments, we know that the adversary $\mathscr{A}$ can obtain private keys for up to $\ell - 1$ linearly independent vectors. We assume w.l.o.g. that $\mathscr{A}$ makes private keys queries for exactly $\ell - 1 = \ell_0 + \ell_1$ independent vectors, which we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell-1} \in \mathbb{Z}_q^\ell$. Among these vectors, we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell_0}$ the vectors queried by $\mathscr{A}$ *b*efore the challenge phase while $\boldsymbol{y}_{\ell_0+1}, \ldots, \boldsymbol{y}_{\ell_0+\ell_1}$ stand for the post-challenge private key queries. In the challenge phase, we denote by $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_\ell^\star) \in \mathbb{Z}_q^\ell$ the message chosen by $\mathscr{A}$. The simulator, given by the four algorithms $(\mathsf{Setup}^\star, \mathsf{Keygen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{Keygen}_1^\star)$, proceeds in the following way.

Setup$^\star(1^\lambda, 1^\ell)$**:** This algorithm is identical to Setup except that h is computed as g^ω for some uniformly random $\omega \leftarrow U(\mathbb{Z}_q)$. Also $\omega = \log_g(h)$ is included in the master secret key. It outputs

$$\mathsf{mpk}^\star := \left(\mathbb{G}, g, h, \{h_i\}_{i=1}^\ell\right).$$

and $\mathsf{msk}^\star = (\omega, \boldsymbol{s}, \boldsymbol{t})$.

Keygen$_0^\star(\mathsf{msk}^\star, \boldsymbol{y})$: This algorithm is used to answer private key queries before the challenge phase and proceeds exactly like Keygen in the real scheme.

Encrypt$^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \mathcal{V}, \{1^{|x_i^\star|}\}_{i=1}^{\ell})$: This algorithm takes as input $\mathsf{mpk}^\star$, $\mathsf{msk}^\star$, the lengths $\{1^{|x_i^\star|}\}_{i=1}^{\ell}$ of all coordinates of $\boldsymbol{x}^\star$ and a set

$$\mathcal{V} = \left\{\{\boldsymbol{y}_j, z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j\rangle, \mathsf{sk}_{\boldsymbol{y}_j}\}_{j=1}^{\ell_0}\right\}$$

containing all pre-challenge independent queries $\{\boldsymbol{y}_j\}_{j=1}^{\ell_0}$, the returned keys and the corresponding linear function evaluations $\{z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j\rangle\}_{j=1}^{\ell_0}$ for the challenge message $\boldsymbol{x}^\star$. The challenge ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ is simulated as follows.

1. Letting $\boldsymbol{z}_{\mathsf{pre}} = (z_1, \ldots, z_{\ell_0})^\top \in \mathbb{Z}_q^{\ell_0}$, compute an arbitrary $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^{\ell}$ such that $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \bmod q$, where

$$\mathbf{Y}_{\mathsf{pre}} = \begin{bmatrix} \boldsymbol{y}_1^\top \\ \vdots \\ \boldsymbol{y}_{\ell_0}^\top \end{bmatrix} \in \mathbb{Z}_q^{\ell_0 \times \ell}.$$

   Note that $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_\ell)^\top$ does not have to be small and can be obtained via Gaussian elimination.

2. Compute the ciphertext by sampling $r, r' \hookleftarrow U(\mathbb{Z}_q)$ uniformly and computing $(C^\star, D^\star) = (g^r, h^{r'})$ as well as

$$E_i^\star = g^{\bar{x}_i} \cdot C^{\star\, s_i} \cdot D^{\star\, t_i} \qquad \forall i \in [\ell].$$

Output the simulated ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ together with the state information $\mathsf{st}' = (\bar{\boldsymbol{x}}, r, r')$.

Keygen$_1^\star(\mathsf{msk}^\star, \boldsymbol{y}, z = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle, \mathsf{st}')$: On input of $\mathsf{msk}^\star = (\omega, \boldsymbol{s}, \boldsymbol{t})$, a post-challenge query $\boldsymbol{y} \in \mathbb{Z}_q^{\ell}$, the evaluation $z = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle$ of the linear function $f_{\boldsymbol{y}}(\boldsymbol{x}^\star)$ on the message $\boldsymbol{x}^\star$ and the state information $\mathsf{st}' = (\bar{\boldsymbol{x}}, r, r') \in \mathbb{Z}_q^{\ell} \times \mathbb{Z}_q^2$, this algorithm computes

$$\begin{aligned} t'_{\boldsymbol{y}} &= \langle \boldsymbol{t}, \boldsymbol{y}\rangle + \frac{1}{\omega \cdot (r' - r)} \cdot (\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - z) \bmod q. \\ s'_{\boldsymbol{y}} &= \langle \boldsymbol{s}, \boldsymbol{y}\rangle - \frac{1}{(r' - r)} \cdot (\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - z) \bmod q. \end{aligned} \tag{5.2}$$

and returns $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$.

Observe that the ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ produced by Encrypt* is distributed in such a way that $(C^\star, D^\star) = (g^r, g^{\omega \cdot (r + (r' - r))})$ and

$$(E_1^\star, \ldots, E_\ell^\star) = g^{\bar{\boldsymbol{x}} + \omega \cdot (r' - r) \cdot \boldsymbol{t}} \cdot (h_1, \ldots, h_\ell)^r,$$

so that, for any $\boldsymbol{y} = (y_1, \ldots, y_\ell)^\top \in \mathbb{Z}_q^\ell$, we have

$$\prod_{i=1}^{\ell} E_i^{\star\, y_i} = g^{\langle \tilde{\boldsymbol{x}}, \boldsymbol{y} \rangle + \omega \cdot (r' - r) \cdot \langle \boldsymbol{t}, \boldsymbol{y} \rangle} \cdot (g^{\langle \boldsymbol{s}, \boldsymbol{y} \rangle} \cdot h^{\langle \boldsymbol{t}, \boldsymbol{y} \rangle})^r,$$

which implies

$$\prod_{i=1}^{\ell} E_i^{\star\, y_i} / (C^{\star\, s'_{\boldsymbol{y}}} \cdot D^{\star\, t'_{\boldsymbol{y}}}) = g^z.$$

This shows that decrypting the simulated ciphertext $(C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ using the simulated key $\mathsf{sk}_{\boldsymbol{y}} = (s'_{\boldsymbol{y}}, t'_{\boldsymbol{y}})$ yields $z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$, as required.

We now proceed to show that the simulation is computationally indistinguishable from the real experiment under the DDH assumption.

The proof uses a sequence of games that begins with a game in which the challenger interacts with the adversary as in real experiment and ends with a game where the challenger interacts with the adversary as in the ideal experiment. For Game_i and Game_j we denote by $\mathsf{Adv}^{ij}_{\mathscr{A}}(\lambda)$ the advantage of a PPT algorithm $\mathscr{A}$ in distinguishing between Game_i and Game_j. Formally the challenger $\mathscr{C}$ flips a coin $b \leftarrow\!\!\$\ \{0,1\}$. If $b = 0$ it interacts with the adversary as in Game_i, else it interacts as in Game_j. At the end of the interaction $\mathscr{A}$ will have to make its guess $b' \in \{0,1\}$. We define $\mathsf{Adv}^{ij}_{\mathscr{A}}(\lambda) := \left|\Pr[b' = b] - \frac{1}{2}\right|$.

Game_0: In this game the challenger interacts with the adversary as in the real experiment.

Game_1: We modify the generation of the ciphertext $C^\star_{\boldsymbol{x}} = (C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$, such that the experiment first computes

$$C^\star = g^r \ \text{ and } \ D^\star = h^r, \tag{5.3}$$

for a randomly sampled $r \leftarrow\!\!\$\ \mathbb{Z}_q$. Then, it uses $\mathsf{msk} := \{s_i, t_i\}_{i=1}^{\ell}$ to compute

$$E_i^\star = g^{x_i^\star} \cdot C^{\star\, s_i} \cdot D^{\star\, t_i}. \tag{5.4}$$

It can be observed that $C^\star_{\boldsymbol{x}} = (C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$ has the same distribution as in Game 0. We hence have $\mathsf{Adv}^{01}_{\mathscr{A}}(\lambda) = 0$.

Game_2: We modify again the generation of $C^\star_{\boldsymbol{x}} = (C^\star, D^\star, E_1^\star, \ldots, E_\ell^\star)$. Namely, instead of computing the pair $(C^\star, D^\star)$ as in (5.3), the experiment samples $r, r' \leftarrow\!\!\$\ U(\mathbb{Z}_q)$ and sets

$$C^\star = g^r \ \text{ and } \ D^\star = h^{r'}.$$

The ciphertext components $(E_1^\star, \ldots, E_\ell^\star)$ are still computed as per (5.4). Under the DDH assumption, this modification should not significantly affect $\mathscr{A}$'s view and we have $\mathsf{Adv}^{12}_{\mathscr{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathscr{B}}(\lambda)$.

Game_3: In this game, the experiment runs exactly as in the ideal case. Lemma 5.2 shows that $\mathsf{Adv}^{23}_{\mathscr{A}}(\lambda) = 0$.

Combining the above, we find

$$|\Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{Real}}_{\mathscr{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{Ideal}}_{\mathscr{A}}(1^\lambda)]| \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathscr{B}}(1^\lambda),$$

as claimed.

□

Lemma 5.2. *The advantage of an adversary $\mathscr{A}$ in distinguishing between* Game_2 *and* Game_3 *is* 0.

Proof. To prove the result, we define the following two variants of these games.

Game'_2: This game is identical to Game_2 except that, at the outset of the game, the challenger chooses a random vector $\mathbf{\Delta x} \leftarrow\!\!\!\!\!\!\rightarrow U(\mathbb{Z}_q^\ell)$. It interacts with $\mathscr{A}$ as in Game_2 until the challenge phase, at which point it computes an arbitrary vector $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^\ell$ satisfying $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \mathbf{Y}_{\mathsf{pre}} \cdot \boldsymbol{x}^\star \bmod q$, where $\mathbf{Y}_{\mathsf{pre}} \in \mathbb{Z}_q^{\ell_0 \times \ell}$ is the matrix whose rows are the first ℓ_0 independent key queries. At this point, the challenger checks whether $\mathbf{\Delta x} = \bar{\boldsymbol{x}} - \boldsymbol{x}^\star \bmod q$ (we call Guess this event). If not, it aborts the interaction with $\mathscr{A}$ and replaces $\mathscr{A}$'s output with 0. Otherwise, it proceeds like Game_2 and outputs whatever $\mathscr{A}$ outputs. Since $\mathbf{\Delta x}$ is drawn uniformly and independently of $\mathscr{A}$'s view, we have $\Pr[\mathsf{Guess}] = 1/q^\ell$.

Game'_3: This game is like Game_3, except that, at the very beginning of the game, the challenger chooses a random $\mathbf{\Delta x} \leftarrow\!\!\!\!\!\!\rightarrow U(\mathbb{Z}_q^\ell)$. It proceeds like Game_3 until the challenge phase, at which point it samples an arbitrary $\bar{\boldsymbol{x}} \in \mathbb{Z}_q^\ell$ satisfying $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \bmod q$. Then, it checks whether $\mathbf{\Delta x} = \bar{\boldsymbol{x}} - \boldsymbol{x}^\star \bmod q$ (we call Guess this event). If not, it aborts and replaces $\mathscr{A}$'s output with 0. Otherwise, it proceeds identically to Game_3 and outputs the same result as $\mathscr{A}$.

Now, we claim that Game'_2 and Game'_3 are identical. To see this, we first note that, conditionally on ¬Guess, both games output 0. If Guess occurs, we observe that Game'_3 is identical to Game'_2 when the master secret key is replaced by $(\boldsymbol{s}', \boldsymbol{t}') \in \mathbb{Z}_q^\ell \times \mathbb{Z}_q^\ell$, where

$$\begin{aligned}
t'_i &= t_i + \frac{1}{\omega \cdot (r' - r)} \cdot \mathbf{\Delta x} \\
&= t_i + \frac{1}{\omega \cdot (r' - r)} \cdot (\bar{x}_i - x^\star_i) \bmod q \qquad \forall i \in [\ell] \\
s'_i &= s_i - \frac{1}{r' - r} \cdot \mathbf{\Delta x} \\
&= s_i - \frac{1}{r' - r} \cdot (\bar{x}_i - x^\star_i) \bmod q.
\end{aligned}$$

Indeed, $(\boldsymbol{s}', \boldsymbol{t}')$ has the same distribution as $(\boldsymbol{s}, \boldsymbol{t})$ conditionally on mpk. By construction, we also have $\langle \boldsymbol{s}', \boldsymbol{y} \rangle = \langle \boldsymbol{s}, \boldsymbol{y} \rangle$ and $\langle \boldsymbol{t}', \boldsymbol{y} \rangle = \langle \boldsymbol{t}, \boldsymbol{y} \rangle$ in all pre-challenge queries $\boldsymbol{y} \in \mathbb{Z}_q^\ell$. Moreover, we have

$$g^{\bar{\boldsymbol{x}} + \omega \cdot (r'-r) \cdot \boldsymbol{t}} \cdot (h_1, \ldots, h_\ell)^r = g^{\boldsymbol{x}^\star + \omega \cdot (r'-r) \cdot \boldsymbol{t}'} \cdot (h_1, \ldots, h_\ell)^r.$$

Finally, answering post-challenge queries $\boldsymbol{y} \in \mathbb{Z}_q^\ell$ using $(\boldsymbol{s}', \boldsymbol{t}')$ gives exactly the distribution (5.2). This implies the games are indeed identical, therefore $\mathsf{Adv}_{\mathscr{A}}^{2'3'}(\lambda) = 0$.

To conclude, notice that any adversary $\mathscr{A}$ that can distinguish between Game_2 and Game_3 can be used to distinguish between Game_2' and Game_3', with a loss factor of q^ℓ in the advantage:

$$\mathsf{Adv}_{\mathscr{A}}^{2'3'}(\lambda) = \frac{1}{q^\ell} \cdot \mathsf{Adv}_{\mathscr{A}}^{23}(\lambda)$$

This holds since the probability that $\mathscr{A}$ outputs the correct bit b' when distinguishing between Game_2' and Game_3' is equal to:

$$\Pr[b' = b] = \Pr[b' = b | \mathsf{Guess}] \cdot \Pr[\mathsf{Guess}] + \Pr[b' = b | \overline{\mathsf{Guess}}] \cdot \Pr[\overline{\mathsf{Guess}}]$$

which is equivalent to:

$$\Pr[b' = b] - \frac{1}{2} = \left(\Pr[b' = b | \mathsf{Guess}] - \frac{1}{2} \right) \cdot \Pr[\mathsf{Guess}]$$

By considering the equality in absolute value, we get the desired relation between the advantages.

□

5.3 Adaptive Simulation Security for Inner Products over $\mathbb{Z}$ from DCR

Another result of [ALMT20], for which we will not give details in this work, constructs a generic compiler that transforms any AD-IND secure IPFE, that satisfies some mild conditions, into an AD-SIM secure IPFE, when the IPFE evaluates inner products mod an integer. Unfortunately, the resulting IPFE scheme has a stateful key generation algorithm. This means that a trusted authority has to keep track of all the keys that are issued. We can apply this general result to the DCR-based scheme from [ALS16], that evaluates inner products over $\mathbb{Z}_N$ to obtain a stateful scheme.

However, in this section we show that a variant of the Paillier-based scheme of Agrawal *et al.* [ALS16], that evaluates inner products over the integers, can also be proved simulation-secure against adaptive adversaries. In this construction, the key generation algorithm is stateless.

5.3.1 Overview

To proof strategy for this result is the same as the one discussed in section 5.2.1. Our variant of the [ALS16] scheme differs from the original in that the master secret keys are no longer sampled from a Gaussian distribution but are rather sampled uniformly in a large interval.

The reason why we need larger master secret keys is that, in the challenge phase, our simulator has to sample a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ that should satisfy an equation of the form $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^{\ell_0}$, for some given $\mathbf{Y}_{\mathsf{pre}} \in \mathbb{Z}^{\ell_0 \times \ell}$ and $\boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^{\ell_0}$, in order to be consistent with responses $\boldsymbol{z}_{\mathsf{pre}} = (z_1, \ldots, z_{\ell_0})$ to all pre-challenge queries. Because we don't know how to compute small integer solutions to the system $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^{\ell_0}$, our simulator can only sample a dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ with large entries. At each post-challenge query $\boldsymbol{y} \in \mathbb{Z}^\ell$, the simulator has to "program" the returned functional secret key in such a way that it decrypts the simulated ciphertext to the value $z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$ dictated by the oracle. For this purpose, the "programmed" key $s'_{\boldsymbol{y}}$ must consist of the sum (over $\mathbb{Z}$) of the real key $s_{\boldsymbol{y}} = \langle \boldsymbol{s}, \boldsymbol{y} \rangle$ and a multiple of the difference $\langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle - z$ between the function evaluation $f_{\boldsymbol{y}}(\bar{\boldsymbol{x}}) = \langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle$ and the oracle value $z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle$. Since $\langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle - z$ may be large over $\mathbb{Z}$, we need to sample the entries of $\boldsymbol{s} \in \mathbb{Z}^\ell$ from a sufficiently wide interval so as to "drown" the statistical discrepancy between the distributions of the master secret $\boldsymbol{s} \in \mathbb{Z}^\ell$ and its shifted variant $\boldsymbol{s}' = \boldsymbol{s} + \gamma \cdot (\bar{\boldsymbol{x}} - \boldsymbol{x}^\star) \in \mathbb{Z}^\ell$ for which $s'_{\boldsymbol{y}} = \langle \boldsymbol{s}', \boldsymbol{y} \rangle$. Since RSA moduli should asymptotically contain $\lambda^3/\mathrm{polylog}(\lambda)$ bits to resist factorization attacks, we need to sample each entry of $\boldsymbol{s} \in \mathbb{Z}^\ell$ from an interval of cardinality $O(2^{\ell^2 \cdot \log \ell + \lambda^3/\mathrm{polylog}(\lambda)})$. Despite somewhat large secret keys, the scheme remains computationally efficient.

5.3.2 The DCR-based Construction

Below we give the description of the IPFE scheme that works for the class of functions of bounded inner-products:

$$\mathscr{F} = \left\{ f_{\boldsymbol{y}} : ([-X, X] \cap \mathbb{Z})^\ell \to \mathbb{Z} : \boldsymbol{y} \in ([-Y, Y] \cap \mathbb{Z})^\ell \text{ and } f_{\boldsymbol{y}}(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle \right\}.$$

Setup$(1^\lambda, 1^\ell, X, Y)$: Choose safe primes $p = 2p' + 1$ and $q = 2q' + 1$ with p', q' also primes, such that $\ell XY < N/2$, where $N = pq$. Sample $g' \leftarrow U(\mathbb{Z}^*_{N^2})$ and set $g := g'^{2N} \bmod N^2$. Next for each $i \in [\ell]$ sample $s_i \leftarrow U([-S, S] \cap \mathbb{Z})$, where $S = 2^{\lambda+\ell-1} \cdot \bar{X}^{\ell+1} \cdot \ell N^2$ and $\bar{X} := X + XY \cdot \ell^2 \cdot (Y\sqrt{\ell})^\ell$ and then compute $h_i = g^{s_i} \bmod N^2$. Define $\mathsf{msk} := \boldsymbol{s} = (s_1, \ldots, s_\ell)^\top \in \mathbb{Z}^\ell$ and $\mathsf{mpk} := \left(N, g, \{h_i\}_{i=1}^\ell, X, Y\right)$

Keygen$(\mathsf{msk}, \boldsymbol{y})$: To generate a secret key from the vector $\boldsymbol{y} \in [-Y, Y]^\ell$ using $\mathsf{msk} = \boldsymbol{s} = (s_1, \ldots, s_\ell)^\top$, compute $s_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle = \sum_{i=1}^\ell s_i \cdot y_i \in \mathbb{Z}$ and return $\mathsf{sk}_{\boldsymbol{y}} := (\boldsymbol{y}, s_{\boldsymbol{y}})$.

Encrypt$(\mathsf{mpk}, \boldsymbol{x})$: Given the public key mpk, to encrypt a message $\boldsymbol{x} \in [-X, X]^\ell$, sample $r \leftarrow U(\{0, 1, \ldots, N/4\})$ and compute

$$c_0 = g^r \bmod N^2, \qquad c_i = (1 + x_i N) \cdot h_i^r \bmod N^2 \qquad \forall i \in [\ell]$$

and output $\mathbf{c} = (c_0, \{c_i\}_{i=1}^\ell) \in \left(\mathbb{Z}_{N^2}^*\right)^{\ell+1}$.

Decrypt$(\mathsf{mpk}, \mathsf{sk}_{\boldsymbol{y}}, \mathbf{c})$: On input of a functional decryption key $\mathsf{sk}_{\boldsymbol{y}} = (\boldsymbol{y}, s_{\boldsymbol{y}})$ and a ciphertext $\mathbf{c} = (c_0, c_1, \ldots, c_\ell)$, compute

$$\mathbf{c}_{\boldsymbol{y}} = c_0^{-s_{\boldsymbol{y}}} \cdot \prod_{i=1}^{\ell} c_i^{y_i} \bmod N^2$$

Then output $\log_{1+N}(\mathbf{c}_{\boldsymbol{y}}) = \frac{\mathbf{c}_{\boldsymbol{y}} - 1 \bmod N^2}{N}$.

*C*orrectness: Suppose that we want to decrypt $\boldsymbol{c} = \{c_i\}_{i=0}^\ell$ using $\mathsf{sk}_{\boldsymbol{y}} = (\boldsymbol{y}, \langle \boldsymbol{s}, \boldsymbol{y} \rangle)$. Observe that we have the following equalities modulo N^2:

$$\prod_{i=1}^{\ell} c_i^{y_i} = \prod_{i=1}^{\ell} (1 + x_i N)^{y_i} \cdot g^{r \cdot s_i y_i} = (1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot g^{r \cdot \langle \boldsymbol{s}, \boldsymbol{y} \rangle} = (1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot c_0^{\langle \boldsymbol{s}, \boldsymbol{y} \rangle},$$

so that $\mathbf{c}_{\boldsymbol{y}} = (1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \bmod N^2$. Recall that $(1+N)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} = 1 + \langle \boldsymbol{x}, \boldsymbol{y} \rangle \cdot N \bmod N^2$, so that computing discrete logarithms in the subgroup generated by $1 + N$ is easy. This enables the computation of $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \bmod N$. By the choice of parameters we have $|\langle \boldsymbol{x}, \boldsymbol{y} \rangle| \leq \ell \cdot \|\boldsymbol{x}\|_\infty \|\boldsymbol{y}\|_\infty \leq \ell \cdot X \cdot Y < N/2$, so we actually recover $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ computed over $\mathbb{Z}$.

5.3.3 The Security Proof

Theorem 5.3. *Under the* DCR *assumption, the above construction achieves adaptive simulation-based security.*

Proof. To prove the theorem we first describe the PPT simulator and show that under the DCR assumption the real experiment is indistinguishable from the ideal experiment.

In both experiments, we know that the adversary $\mathscr{A}$ can obtain private keys for up to $\ell - 1$ linearly independent vectors over $\mathbb{Z}$, a limit that is intrinsic to the functionality. We assume w.l.o.g. that $\mathscr{A}$ makes private keys queries for exactly $\ell - 1 = \ell_0 + \ell_1$ independent vectors, which we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell-1} \in \mathbb{Z}_q^\ell$. Among these vectors, we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell_0}$ the vectors queried by $\mathscr{A}$ *b*efore the challenge phase while $\boldsymbol{y}_{\ell_0+1}, \ldots, \boldsymbol{y}_{\ell_0+\ell_1}$ stand for the post-challenge private key queries. In the challenge phase, we denote by $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_\ell^\star) \in \mathbb{Z}_q^\ell$ the message chosen by $\mathscr{A}$. The simulator, given by $(\mathsf{Setup}^\star, \mathsf{Keygen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{Keygen}_1^\star)$, proceeds as follows.

Setup$^\star(1^\lambda, 1^\ell, X, Y)$: This algorithm chooses safe primes $p = 2p'+1$ and $q = 2q'+1$ such that $\ell XY < N/2$, and sets $N = pq$. It samples $g' \leftarrow U(\mathbb{Z}^*_{N^2})$ and sets $g := g'^{2N} \bmod N^2$. Next, for each $i \in [\ell]$, it samples $s_i \leftarrow U([-S, S] \cap \mathbb{Z})$, where $S = 2^{\lambda+\ell-1} \cdot \bar{X}^{\ell+1} \cdot \ell N^2$ and $\bar{X} := X + XY \cdot \ell^2 \cdot (Y\sqrt{\ell})^\ell$, and computes $h_i = g^{s_i} \bmod N^2$. It defines the master secret key $\mathsf{msk}^\star = \big(\boldsymbol{s}, p, q\big)$, where $\boldsymbol{s} = (s_1, \ldots, s_\ell)^\top$, and the master public key $\mathsf{mpk}^\star = \big(N, g, \{h_i\}_{i=1}^\ell, X, Y\big)$

Keygen$^\star_0(\mathsf{msk}^\star, \boldsymbol{y})$: This algorithm is used to generate all the pre-challenge functional decryption queries. To generate a secret key for $\boldsymbol{y} \in [-Y, Y]^\ell$, it computes and outputs $\mathsf{sk}_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle = \sum_{i=1}^\ell s_i \cdot y_i \in \mathbb{Z}$.

Encrypt$^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \{(\boldsymbol{y}_1, z_1), (\boldsymbol{y}_2, z_2), \ldots, (\boldsymbol{y}_{\ell_0}, z_{\ell_0})\})$: Given $\mathsf{mpk}^\star$, $\mathsf{msk}^\star$ and all the pre-challenge pairs $(\boldsymbol{y}_j, z_j) \in [-Y, Y]^\ell \times \mathbb{Z}$, where $z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j \rangle \in \mathbb{Z}$ and $\boldsymbol{x}^\star$ is the challenge message, it first computes a 'small' dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ such that $\langle \bar{\boldsymbol{x}}, \boldsymbol{y}_j \rangle = z_j$ for all $j \in [\ell_0]$, as follows.

Letting $\boldsymbol{z}_{\mathsf{pre}} = (z_1, \ldots, z_{\ell_0})^\top \in \mathbb{Z}^{\ell_0}$, it computes $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ as in Lemma 2.15, such that $\|\bar{\boldsymbol{x}}\|_\infty \leq \|\boldsymbol{z}_{\mathsf{pre}}\|_\infty \cdot \ell \cdot (Y\sqrt{\ell_0})^{\ell_0}$ and $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^{\ell_0}$, where

$$\mathbf{Y}_{\mathsf{pre}} = \begin{bmatrix} \boldsymbol{y}_1^\top \\ \vdots \\ \boldsymbol{y}_{\ell_0}^\top \end{bmatrix} \in \mathbb{Z}^{\ell_0 \times \ell},$$

Next, it samples $a \leftarrow U(\mathbb{Z}^*_N)$ and $b \leftarrow U(\mathbb{Z}_{N'})$, where $N' = p'q'$, and computes

$$c_0^\star = (1 + aN) \cdot g^b \bmod N^2, \qquad c_i^\star = (1 + \bar{x}_i N) \cdot (c_0^\star)^{s_i} \bmod N^2 \qquad \forall i \in [\ell].$$

It outputs the simulated ciphertext $\mathbf{c}^\star = \big(c_0^\star, \{c_i^\star\}_{i=1}^\ell\big) \in \Big(\mathbb{Z}^*_{N^2}\Big)^{\ell+1}$ together with the state information $\mathsf{st} := \big(\bar{\boldsymbol{x}}, a, N'\big)$

Keygen$^\star_1(\mathsf{msk}^\star, (\boldsymbol{y}, z = \langle \boldsymbol{y}, \boldsymbol{x}^\star \rangle), \mathsf{st})$: Post-challenge key queries are handled as follows. Upon receiving a pair $(\boldsymbol{y}, z = \langle \boldsymbol{x}^\star, \boldsymbol{y} \rangle)$, it first computes $u, v \in \mathbb{Z}$ such that $uN + vN' = 1$ and $\gamma := (a^{-1} \bmod N) \cdot vN' \bmod NN'$ then computes and outputs

$$s'_{\boldsymbol{y}} := \langle \boldsymbol{s}, \boldsymbol{y} \rangle + \gamma \cdot (\langle \bar{\boldsymbol{x}}, \boldsymbol{y} \rangle - z) \in \mathbb{Z}.$$

In order to prove that the real experiment is computationally indistinguishable from the ideal one, we use a sequence of games. We denote by $\mathsf{Adv}^{ij}_{\mathscr{A}}(\lambda)$ the advantage of an adversary $\mathscr{A}$ in distinguishing between Game_i and Game_j. More precisely, a challenger $\mathscr{C}$ flips a coin $b \leftarrow \{0,1\}$. If $b = 0$ the challenger interacts with the adversary $\mathscr{A}$ as in Game_i while, if $b = 1$, it interacts as in Game_j. At the end of the interaction, $\mathscr{A}$ outputs $b' \in \{0,1\}$. The advantage is defined as $\mathsf{Adv}^{ij}_{\mathscr{A}}(\lambda) := \big|\Pr[b' = b] - \frac{1}{2}\big|$.

Game$_0$: This is the real experiment in which the challenger generates the parameters and interacts with the adversary as in the real experiment.

Game$_1$: This game is exactly as the previous one except that the challenge ciphertext is computed as follows: $r \leftarrow U(\{0,1,\ldots,N/4\})$ is sampled and

$$c_0^\star = g^r \bmod N^2, \qquad c_i^\star = (1 + x_i^\star N) \cdot (c_0^\star)^{s_i} \bmod N^2, \text{ for } i \in [\ell]$$

This is possible since the challenger knows the secret key $\mathsf{msk} = \left(\{s_i\}_{i=1}^{\ell}\right)$. Notice that Game_0 is identical to Game_1. So, $\mathsf{Adv}^{01}_{\mathscr{A}}(\lambda) = 0$.

Game$_2$: In this experiment, the computation of $c_0^\star$ is modified. In the challenge phase, the challenger samples $r \leftarrow U(\mathbb{Z}_{N'})$, where $N' = p'q'$, and computes $c_0^\star := g^r \bmod N^2$. By Lemma 2.12, the statistical distance between distributions $U(\{0,1,2,\ldots,N/4\}) \bmod N'$ and $U(\mathbb{Z}_{N'})$ is $< \frac{1}{p} + \frac{1}{q}$, which is negligible. Hence, Game_1 and Game_2 are statistically indistinguishable. More precisely, we have $\mathsf{Adv}^{12}_{\mathscr{A}}(\lambda) < 1/p + 1/q$.

Game$_3$: This experiment is like the previous one, except that $c_0^\star$ is generated by sampling $t \leftarrow\!\!\!\rightarrow U(\mathbb{Z}^*_{N^2})$ and computing $c_0^\star := t^2 \bmod N^2$. Under the DCR assumption, Game_2 and Game_3 are computationally indistinguishable. Indeed, in Game_2, as long as g has order N', the distribution $\{g^r \mid r \leftarrow\!\!\!\rightarrow U(\mathbb{Z}_{N'})\}$ is the uniform distribution in the subgroup of $2N$-th residues. The DCR assumption implies that the latter distribution is computationally indistinguishable from the distribution $\{t^2 \bmod N^2 \mid t \leftarrow\!\!\!\rightarrow U(\mathbb{Z}^*_{N^2})\}$. Since a random $2N$-th residue g generates the entire subgroup of $2N$-th residues with probability $\frac{\varphi(N')}{N'} = 1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{N'}$, we obtain

$$\left(1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{N'}\right) \cdot \mathsf{Adv}^{23}_{\mathscr{A}}(\lambda) \le \mathsf{Adv}^{\mathsf{DCR}}_{\mathscr{B}}(\lambda).$$

Game$_4$: In this game, we sample $a \leftarrow U(\mathbb{Z}^*_N)$ and $b \leftarrow U(\mathbb{Z}_{N'})$ and compute $c_0^\star := (1 + aN) \cdot g^b \bmod N^2$. Observe that $\{t^2 \bmod N^2 \mid t \leftarrow\!\!\!\rightarrow U(\mathbb{Z}^*_{N^2})\}$ is the same as the distribution $\{(1 + \alpha N) \cdot g^\beta \bmod N^2 \mid \alpha \leftarrow\!\!\!\rightarrow U(\mathbb{Z}_N),\ \beta \leftarrow\!\!\!\rightarrow U(\mathbb{Z}_{N'})\}$. Therefore the statistical distance between the view of the adversary in Game_3 and Game_4 is bounded by $\Delta(a,\alpha) < \frac{1}{p} + \frac{1}{q}$. So, these games are statistically indistinguishable and $\mathsf{Adv}^{34}_{\mathscr{A}}(\lambda) < 1/p + 1/q$.

Game$_5$: This is the ideal experiment where the adversary interacts with the simulator. Lemma 5.4 shows that Game_5 and Game_4 are statistically indistinguishable, which yields the stated result.

Putting the above altogether, we obtain that a PPT adversary $\mathscr{A}$ that can distinguish between the real and the ideal experiment implies an efficient DCR distinguisher $\mathscr{B}$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Real\text{-}Ideal}}_{\mathscr{A}}(1^\lambda) &= |\Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{Real}}_{\mathscr{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Exp}^{\mathsf{Ideal}}_{\mathscr{A}}(1^\lambda)]| \\ &\leq \frac{N'}{\varphi(N')} \cdot \mathsf{Adv}^{\mathsf{DCR}}_{\mathscr{B}}(1^\lambda) + \frac{2}{p} + \frac{2}{q} + 2^{-\lambda}.\end{aligned}$$

□

Lemma 5.4. *The advantage of any distinguisher between* Game_4 *and* Game_5 *is statistically negligible and* $\mathsf{Adv}^{45}_{\mathscr{A}}(\lambda) \leq 2^{-\lambda}$.

Proof. In order to prove the claim, we simultaneously define Game'_4 and Game'_5 as follows. For each $k \in \{4,5\}$, define Game'_k identically to Game_k except that, at the outset of the game, the challenger samples $\Delta\boldsymbol{x} \hookleftarrow U([-\bar{X},\bar{X}]^\ell)$, where $\bar{X} = X + XY \cdot \ell^2 \cdot (Y\sqrt{\ell})^\ell$. Before generating the challenge ciphertext, the challenger uses Lemma 2.15 to compute $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ such that $\|\bar{\boldsymbol{x}}\|_\infty \leq \ell XY \cdot \ell (Y\sqrt{\ell})^\ell$ and $\mathbf{Y}_{\mathsf{pre}} \cdot \bar{\boldsymbol{x}} = \mathbf{Y}_{\mathsf{pre}} \cdot \boldsymbol{x}^\star$, where $\mathbf{Y}_{\mathsf{pre}}$ is the matrix obtained by stacking up the (linearly independent) transposed vectors $\boldsymbol{y}^\top$ occurring in pre-challenge queries. If $\Delta\boldsymbol{x} = \bar{\boldsymbol{x}} - \boldsymbol{x}^\star$ (we call this event Guess), the challenger proceeds as in Game_k. Otherwise, the challenger aborts the game and replaces $\mathscr{A}$'s output b' by a random bit. We claim that any adversary $\mathscr{A}$ that can distinguish between Game_4 and Game_5 with advantage $\mathsf{Adv}^{45}_{\mathscr{A}}(\lambda)$ can be used to distinguish between Game'_4 and Game'_5 with advantage

$$\mathsf{Adv}^{4'5'}_{\mathscr{A}}(\lambda) = \frac{1}{(2\bar{X})^\ell} \cdot \mathsf{Adv}^{45}_{\mathscr{A}}(\lambda). \tag{5.5}$$

Indeed, the probability that $\mathscr{A}$ outputs the correct bit b' when distinguishing between Game'_4 and Game'_5 is equal to

$$\Pr[b' = b] = \Pr[b' = b | \mathsf{Guess}] \cdot \Pr[\mathsf{Guess}] + \Pr[b' = b | \overline{\mathsf{Guess}}] \cdot \Pr[\overline{\mathsf{Guess}}]$$

which is equivalent to

$$\Pr[b' = b] - \frac{1}{2} = \left(\Pr[b' = b | \mathsf{Guess}] - \frac{1}{2}\right) \cdot \Pr[\mathsf{Guess}]$$

By considering the equality in absolute value, we obtain (5.7).

Next, we claim that $\mathsf{Adv}^{4'5'}_{\mathscr{A}}(\lambda) \leq (2\bar{X})^{-\ell} \cdot 2^{-\lambda}$, which implies that Game_4 and Game_5 are indistinguishable. To see this, observe that, when Guess occurs, Game'_5 is identical to a modification of Game'_4 where the master secret key has been replaced by

$$s'_i = s_i + \gamma \cdot \Delta\boldsymbol{x}_i \in \mathbb{Z}, \qquad \forall i \in [\ell]$$

where $\gamma = (a^{-1} \bmod N) \cdot vN' \bmod NN'$ is determined by the Bézout coefficient v for which $uN + vN' = 1$ (and thus $vN' = 1 \bmod N$) and the element $a \in \mathbb{Z}_N^*$ which

used to compute $c_0^\star = (1+aN)\cdot g^b \bmod N^2$ in the challenge ciphertext. (Note that a and v can be chosen by the challenger at the beginning of the game, so that we can define a game where the challenger uses $\{s'_i\}_i$ instead of $\{s_i\}_i$). With this new master secret key $\boldsymbol{s}' = (s'_1, \ldots, s'_\ell)$, notice that the public key remains invariant $h_i = g^{s_i} = g^{s'_i} \bmod N^2$ for all $i \in [\ell]$ and $\langle \boldsymbol{s}, \boldsymbol{y}\rangle = \langle \boldsymbol{s}', \boldsymbol{y}\rangle$ for all pre-challenge queries $\boldsymbol{y} \in \mathbb{Z}^\ell$. We thus obtain

$$\mathsf{Adv}^{4'5'}_{\mathscr{A}}(\lambda) \leq \Delta(\boldsymbol{s}', \boldsymbol{s}) \leq (2\bar{X})^{-\ell} \cdot 2^{-\lambda},$$

where the last inequality follows from the fact that

$$\Delta(\boldsymbol{s}', \boldsymbol{s}) \leq \sum_{i=1}^{\ell} \Delta(s'_i, s_i) \overset{\text{Lemma 2.13}}{\leq} \ell \cdot \frac{\|\gamma \cdot \Delta\boldsymbol{x}\|_\infty}{2S} \leq \frac{NN' \cdot \bar{X}}{2^{\lambda+\ell} \cdot \bar{X}^{\ell+1} \cdot N^2} \leq (2\bar{X})^{-\ell} \cdot 2^{-\lambda}.$$

□

5.4 Adaptive Simulation IPFE from LWE

As already mentioned in Section 5.3, another result of [ALMT20] gives an AD-SIM IPFE compiler, for constructions that evaluate inner products mod an integer. This can be applied to the [ALS16] LWE-based construction for inner products over $\mathbb{Z}_p$, to obtain a stateful AD-SIM secure IPFE, that evaluates inner products mod p. We will not give the details for the stateful construction in this work, but the results can be found in the [ALMT20] paper.

However, in this section we recall a variant of the LWE-based IPFE scheme of [ALS16], that evaluates inner products over the integers, for which we prove AD-SIM security. The key generation algorithm for this scheme is stateless.

5.4.1 Overview

The security proof follows the same pattern as the DDH and DCR-based proofs from the previous sections. The difference in this case is that, shifting the master secret key is not as straightforward as before. The reason is that the decryption process of the LWE-based scheme is different, in the following sense. When we decrypt the ciphertext corresponding to the message vector $\boldsymbol{x} \in \mathbb{Z}^\ell$, computed under the public key $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{U} = \mathbf{A}\cdot\mathbf{R} \in \mathbb{Z}_q^{n\times \ell}$,

$$\begin{aligned} \mathbf{c}_0 &= \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}_0 \in \mathbb{Z}_q^m \\ \mathbf{c}_1 &= \mathbf{U}^\top \cdot \mathbf{s} + \mathbf{e}_1 + \boldsymbol{x} \cdot \lfloor q/K \rfloor \in \mathbb{Z}_q^\ell, \end{aligned}$$

using a functional decryption key $\mathbf{r}_{\boldsymbol{y}} = \mathbf{R}\cdot\boldsymbol{y} \in \mathbb{Z}^m$, we must compute $\boldsymbol{y}^\top \cdot \mathbf{c}_1 - \mathbf{r}_{\boldsymbol{y}}^\top \mathbf{c}_0 = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle \cdot \lfloor q/K \rfloor + \text{small error}$. When we do the same computation, using a shifted functional decryption key of the form $\mathbf{r}'_{\boldsymbol{y}} := \mathbf{r}_{\boldsymbol{y}} + \boldsymbol{\alpha} \cdot \langle \Delta\boldsymbol{x}, \boldsymbol{y}\rangle$, corresponding to a master secret key, of the form $\mathbf{R}' := \mathbf{R} + \boldsymbol{\alpha}$, the extra term $\boldsymbol{\alpha}^\top \cdot \mathbf{c}_0$ appears,

which prevents the correct programming of the post-challenge keys. To solve this problem, during the simulation, we generate a trapdoor matrix corresponding to $\begin{bmatrix}\mathbf{A}\\ \mathbf{c}_0^\top\end{bmatrix} \in \mathbb{Z}_q^{(n+1)\times m}$. The trapdoor allows us to compute a small matrix $\boldsymbol{\alpha} \in \mathbb{Z}^{m\times \ell}$ such that $\mathbf{A}\cdot\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{\alpha}^\top \cdot \mathbf{c}_0 = \texttt{constant}$. This implies that the public key $\mathbf{U} = \mathbf{AR} = \mathbf{AR}'$, remains invariant under the shift $\mathbf{R}' = \mathbf{R} + \boldsymbol{\alpha}$ and also it enables us to correctly program the post-challenge key queries. We can actually choose $\boldsymbol{\alpha} = \Delta \boldsymbol{x}^\top \otimes \mathbf{w}$ of this form, which guarantees that the pre-challenge keys remain invariant to the shift.

To argue that the statistical distance between the master secret key $\mathbf{R}$ and its shift $\mathbf{R}' = \mathbf{R} + \boldsymbol{\alpha}$ is negligible, we use the noise flood Lemma 2.9. Because of this, we need an exponentially large modulus q.

5.4.2 The LWE-based IPFE Construction

Setup$(1^\lambda, 1^\ell, X, Y)$**:** The public parameters are $\Gamma = \{m, n, q, \ell, \alpha, \sigma, \sigma_1, X, Y, K\}$, for $K := \ell \cdot XY$, $m = \Theta(n \log q)$, $\alpha \in (0,1)$ such that $\alpha q = \Omega(\sqrt{n})$, $\sigma = 2^\lambda \cdot \mathsf{poly}(\lambda)$, $\sigma_1 = 2^\lambda \cdot \sigma \cdot \mathsf{poly}(\lambda)$ and $(\sigma_1 + m \cdot \sigma \cdot \alpha q) \cdot Y\sqrt{\ell} < q/2K$.

The public key is generated via the following steps. Sample $\mathbf{A} \leftarrow\!\!\$\ U(\mathbb{Z}_q^{n\times m})$, $\mathbf{R} \leftarrow\!\!\$\ D_{\mathbb{Z}^{m\times\ell},\sigma}$ and compute $\mathbf{U} := \mathbf{AR} \in \mathbb{Z}_q^{n\times\ell}$. Set $\mathsf{mpk} := (\mathbf{A}, \mathbf{U})$ and $\mathsf{msk} := \mathbf{R}$.

Keygen$(\mathsf{msk}, \boldsymbol{y})$**:** On input the $\mathsf{msk} = \mathbf{R} \in \mathbb{Z}^{m\times\ell}$ and a vector $\boldsymbol{y} \in \mathbb{Z}^\ell$ such that $\|\boldsymbol{y}\|_\infty \le Y$, compute $\mathbf{r}_{\boldsymbol{y}} := \mathbf{R}\cdot\boldsymbol{y} \in \mathbb{Z}^m$ and return $\mathsf{sk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathbf{r}_{\boldsymbol{y}})$.

Encrypt$(\mathsf{mpk}, \boldsymbol{x})$**:** To encrypt a message $\boldsymbol{x} \in \mathbb{Z}^\ell$ such that $\|\boldsymbol{x}\|_\infty \le X$, first sample $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n$, $\mathbf{e}_0 \leftarrow\!\!\$\ D_{\mathbb{Z}^m,\alpha q}$, $\mathbf{e}_1 \leftarrow\!\!\$\ D_{\mathbb{Z}^\ell,\sigma_1}$ compute:

$$\begin{aligned}\mathbf{c}_0 &= \mathbf{A}^\top\cdot\mathbf{s} + \mathbf{e}_0 \in \mathbb{Z}_q^m\\ \mathbf{c}_1 &= \mathbf{U}^\top\cdot\mathbf{s} + \mathbf{e}_1 + \boldsymbol{x}\cdot\lfloor q/K\rfloor \in \mathbb{Z}_q^\ell\end{aligned}$$

and return the ciphertext $\mathbf{c} := (\mathbf{c}_0, \mathbf{c}_1) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$.

Decrypt$(\mathsf{sk}_{\boldsymbol{y}}, \mathbf{c})$**:** Given the secret key $\mathsf{sk}_{\boldsymbol{y}} = (\boldsymbol{y}, \mathbf{r}_{\boldsymbol{y}}) \in \mathbb{Z}^{\ell+m}$ and a ciphertext $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1)$ compute $\mathbf{v} := \boldsymbol{y}^\top\cdot\mathbf{c}_1 - \mathbf{r}_{\boldsymbol{y}}^\top\mathbf{c}_0 \bmod q$. Return the vector $\boldsymbol{z} \in \mathbb{Z}^\ell$ such that $|v_i - z_i\cdot\lfloor q/K\rfloor|$ is minimized for all $i \in [\ell]$.

Lemma 5.5 (Correctness)**.** *If* $\sigma_1 = \omega(\sqrt{\log\ell})$, $\alpha q = \omega(\sqrt{\log m})$ *and* $(\sigma_1 + m\cdot\sigma\cdot\alpha q)\cdot Y\sqrt{\ell} < q/2K$ *then for any* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, X, Y)$*, any* $\boldsymbol{x} \in [-X, X]^\ell \cap \mathbb{Z}^\ell$ *and* $\boldsymbol{y} \in [-Y, Y]^\ell \cap \mathbb{Z}^\ell$ *and* $\mathsf{sk}_{\boldsymbol{y}} \leftarrow \mathsf{Keygen}(\mathsf{msk}, \boldsymbol{y})$ *we have*

$$\mathsf{Decrypt}(\mathsf{sk}_{\boldsymbol{y}}, \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x})) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle,$$

with probability exponentially close to 1.

Proof. Suppose that $(\mathbf{c}_0, \mathbf{c}_1) \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \boldsymbol{x})$ for some integer vector $\boldsymbol{x} \in [-X, X]^\ell$. The decryption algorithm computes $\boldsymbol{y}^\top \cdot \mathbf{c}_1 - \mathbf{r}_{\boldsymbol{y}}^\top \mathbf{c}_0 = \langle \mathbf{e}_1 - \mathbf{R}^\top \mathbf{e}_0, \boldsymbol{y} \rangle + \lfloor q/K \rfloor \cdot \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Notice that the condition $|\langle \mathbf{e}_1 - \mathbf{R}^\top \mathbf{e}_0, \boldsymbol{y} \rangle| < q/2K$ ensures that decryption correctly recovers $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. It thus suffices to prove the above inequality.

$$|\langle \mathbf{e}_1 - \mathbf{R}^\top \mathbf{e}_0, \boldsymbol{y} \rangle| \leq \big(\|\mathbf{e}_1\| + \|\mathbf{R}^\top \mathbf{e}_0\|\big) \cdot \|\boldsymbol{y}\| \leq \left(\|\mathbf{e}_1\| + \sqrt{\ell} \cdot \max_{i \in [\ell]} |\mathbf{r}_i^\top \cdot \mathbf{e}_0|\right) \cdot \|\boldsymbol{y}\|,$$

where $\mathbf{r}_i^\top \in \mathbb{Z}^{1 \times m}$ are the rows of $\mathbf{R}^\top$.

By Lemma 2.1 we can bound the Euclidean norm of the Gaussian vectors $\|\mathbf{e}_0\| \leq \alpha q \sqrt{m}$, $\|\mathbf{e}_1\| \leq \sigma_1 \sqrt{\ell}$ and $\|\mathbf{r}_i\| \leq \sigma \sqrt{m}$ respectively. This implies

$$|\langle \mathbf{e}_1 - \mathbf{R}^\top \mathbf{e}_0, \boldsymbol{y} \rangle| \leq (\sigma_1 + m \cdot \sigma \cdot \alpha q) \cdot Y \sqrt{\ell} < q/2K.$$

The last inequality follows from the choice of parameters. □

5.4.3 Security

Theorem 5.6. *The above construction provides AD-SIM security under the LWE assumption.*

Proof. In order to prove the result, we first describe the simulator given by the PPT algorithms $(\mathsf{Setup}^\star, \mathsf{Keygen}_0^\star, \mathsf{Encrypt}^\star, \mathsf{Keygen}_1^\star)$. In a second step, we will prove that the LWE assumption implies the adversary's inability to distinguish the real experiment from the ideal.

Both in the Real and the Ideal experiments, we know that the adversary $\mathscr{A}$ can obtain private keys for up to $\ell - 1$ linearly independent vectors over $\mathbb{Z}$, a limit that is intrinsic to the functionality. We assume w.l.o.g. that $\mathscr{A}$ makes private keys queries for exactly $\ell - 1 = \ell_0 + \ell_1$ independent vectors, which we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell-1} \in \mathbb{Z}_q^\ell$. Among these vectors, we denote by $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{\ell_0}$ the vectors queried by $\mathscr{A}$ *b*efore the challenge phase while $\boldsymbol{y}_{\ell_0+1}, \ldots, \boldsymbol{y}_{\ell_0+\ell_1}$ stand for the post-challenge private key queries. In the challenge phase, we denote by $\boldsymbol{x}^\star = (x_1^\star, \ldots, x_\ell^\star) \in \mathbb{Z}_q^\ell$ the message chosen by $\mathscr{A}$. The simulator proceeds as follows.

$\mathsf{Setup}^\star(1^\lambda, 1^\ell, X, Y)$: The public parameters $\Gamma = \{m, n, q, \ell, \alpha, \sigma, \sigma_1, X, Y, K\}$, are chosen by the experiment as in the descryption of the scheme.

Next it runs $(\bar{\mathbf{A}}, \mathbf{T}_{\bar{\mathbf{A}}}) \leftarrow \mathsf{TrapGen}(1^m, 1^{n+1})$ and parses $\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{a}_0^\top \end{bmatrix} \in \mathbb{Z}_q^{(n+1) \times m}$ and it uses the trapdoor $\mathbf{T}_{\bar{\mathbf{A}}} \in \mathbb{Z}^{m \times m}$ to sample a "small" vector $\mathbf{w} \in \mathbb{Z}^m$ (as in Remark 1) such that

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{a}_0^\top \end{bmatrix} \cdot \mathbf{w} = \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/K \rfloor \end{bmatrix} \in \mathbb{Z}_q^{n+1},$$

Then, it samples $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m \times \ell}, \sigma}$, computes $\mathbf{U} = \mathbf{AR}$ and sets $\mathsf{mpk}^\star = (\mathbf{A}, \mathbf{U})$ and $\mathsf{msk}^\star = (\mathbf{R}, \mathbf{T}_{\bar{\mathbf{A}}}, \mathbf{a}_0, \mathbf{w})$.

Keygen$_0^\star(\mathsf{msk}^\star, \boldsymbol{y})$: On input the $\mathsf{msk} = \mathbf{R} \in \mathbb{Z}^{m\times\ell}$ and a vector $\boldsymbol{y} \in \mathbb{Z}^\ell$ such that $\|\boldsymbol{y}\|_\infty \le Y$, compute $\mathbf{r}_{\boldsymbol{y}} := \mathbf{R}\cdot\boldsymbol{y} \in \mathbb{Z}^m$ and return $\mathsf{sk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathbf{r}_{\boldsymbol{y}})$.

Encrypt$^\star(\mathsf{mpk}^\star, \mathsf{msk}^\star, \{(\boldsymbol{y}_1, z_1), (\boldsymbol{y}_2, z_2), \ldots, (\boldsymbol{y}_{\ell_0}, z_{\ell_0})\})$: Given $\mathsf{mpk}^\star$, $\mathsf{msk}^\star$ and all the pre-challenge pairs $(\boldsymbol{y}_j, z_j) \in [-Y, Y]^\ell \times \mathbb{Z}$, where $z_j = \langle \boldsymbol{x}^\star, \boldsymbol{y}_j\rangle \in \mathbb{Z}$ and $\boldsymbol{x}^\star$ is the challenge message, it first computes a "small" dummy message $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ such that $\langle \bar{\boldsymbol{x}}, \boldsymbol{y}_j\rangle = z_j$ for all $j \in [\ell_0]$, as follows.

Letting $\boldsymbol{z}_{\mathsf{pre}} = (z_1, \ldots, z_{\ell_0})^\top \in \mathbb{Z}^{\ell_0}$, it computes $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ as in Lemma 2.15, such that $\|\bar{\boldsymbol{x}}\|_\infty \le \|\boldsymbol{z}_{\mathsf{pre}}\|_\infty \cdot \ell \cdot (Y\sqrt{\ell_0})^{\ell_0}$ and $\mathbf{Y}_{\mathsf{pre}}\cdot\bar{\boldsymbol{x}} = \boldsymbol{z}_{\mathsf{pre}} \in \mathbb{Z}^{\ell_0}$, where

$$\mathbf{Y}_{\mathsf{pre}} = \begin{bmatrix} \boldsymbol{y}_1^\top \\ \vdots \\ \boldsymbol{y}_{\ell_0}^\top \end{bmatrix} \in \mathbb{Z}^{\ell_0\times\ell},$$

Next, it samples $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$, $\mathbf{e}_1^\star \hookleftarrow D_{\mathbb{Z}^\ell,\sigma_1}$ and computes the ciphertext as:

$$\begin{aligned} \mathbf{c}_0^\star &= \mathbf{a}_0 \\ \mathbf{c}_1^\star &= \mathbf{R}^\top\cdot\mathbf{c}_0^\star + \mathbf{e}_1^\star + \bar{\boldsymbol{x}}\cdot\lfloor q/K\rfloor \in \mathbb{Z}_q^\ell \end{aligned}$$

It outputs the simulated ciphertext $\mathbf{c}^\star = (\mathbf{c}_0^\star, \mathbf{c}_1^\star)$.

Keygen$_1^\star(\mathsf{msk}^\star, (\boldsymbol{y}, z = \langle \boldsymbol{y}, \boldsymbol{x}^\star\rangle), \mathsf{st})$: Post-challenge key queries as answered as follows. Upon receiving a pair $(\boldsymbol{y}, z = \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle)$, it computes $\mathbf{r}'_{\boldsymbol{y}} := \mathbf{R}\boldsymbol{y} + (\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - z)\cdot\mathbf{w} \in \mathbb{Z}^m$, and outputs $\mathsf{sk}_{\boldsymbol{y}} := (\boldsymbol{y}, \mathbf{r}'_{\boldsymbol{y}})$.

Notice that keys $\mathsf{sk}_{\boldsymbol{y}}$ corresponding to post-challenge queries decrypt to $\langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle$.

$$\begin{aligned} \boldsymbol{y}^\top\cdot\mathbf{c}_1^\star - \mathbf{r}'^\top_{\boldsymbol{y}}\cdot\mathbf{c}_0^\star &= (\mathbf{R}\boldsymbol{y})^\top\cdot\mathbf{c}_0^\star + \langle \mathbf{e}_1^\star, \boldsymbol{y}\rangle + \lfloor q/K\rfloor\cdot\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - \\ &- (\mathbf{R}\boldsymbol{y})^\top\cdot\mathbf{c}_0^\star - \big(\langle \bar{\boldsymbol{x}}, \boldsymbol{y}\rangle - z\big)\cdot\mathbf{w}^\top\cdot\mathbf{c}_0^\star \\ &= \langle \mathbf{e}_1^\star, \boldsymbol{y}\rangle + \langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle\cdot\lfloor q/K\rfloor \end{aligned}$$

So the above decrypts to $\langle \boldsymbol{x}^\star, \boldsymbol{y}\rangle$ since $|\langle \mathbf{e}_1^\star, \boldsymbol{y}\rangle| < q/2K$ (Correctness Lemma 5.5).

In order to prove that the real experiment is computationally indistinguishable from the ideal experiment, we use the following sequence of games.

Game 0: This the real AD-SIM experiment. The pair $(\mathsf{mpk}, \mathsf{msk})$ is generated as in the real experiment, namely $\mathbf{A} \hookleftarrow \mathbb{Z}_q^{n\times m}$, $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m\times\ell},\sigma}$ and $\mathbf{U} = \mathbf{AR} \in \mathbb{Z}_q^{n\times\ell}$. The public and the secret keys are set as $\mathsf{mpk} := (\mathbf{A}, \mathbf{U})$, $\mathsf{msk} := \mathbf{R}$. To encrypt the adversarially-chosen $\boldsymbol{x}^\star \in [-X, X]^\ell \cap \mathbb{Z}^\ell$, the the experiment samples $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$, $\mathbf{e}_1^\star \hookleftarrow D_{\mathbb{Z}^\ell,\sigma_1}$ computes

$$\begin{aligned} \mathbf{c}_0^\star &= \mathbf{A}^\top\cdot\mathbf{s} + \mathbf{e}_0^\star \in \mathbb{Z}_q^m \\ \mathbf{c}_1^\star &= \mathbf{U}^\top\cdot\mathbf{s} + \mathbf{e}_1^\star + \boldsymbol{x}^\star\cdot\lfloor q/K\rfloor \in \mathbb{Z}_q^\ell \end{aligned}$$

and sets $\mathbf{c}^\star := (\mathbf{c}_0^\star, \mathbf{c}_1^\star) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^\ell$. To answer both the pre-challenge and the post-challenge key queries for some vector $\boldsymbol{y} \in [-Y, Y]^\ell \cap \mathbb{Z}^\ell$, the experiment returns $(\boldsymbol{y}, \mathbf{r}_{\boldsymbol{y}})$, where $\mathbf{r}_{\boldsymbol{y}} := \mathbf{R} \cdot \boldsymbol{y} \in \mathbb{Z}^m$, exactly as in the real experiment.

Game 1: This game differs from the previous one in that the ciphertext is computed as: $\mathbf{s} \hookleftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$, $\mathbf{e}_1^\star \hookleftarrow D_{\mathbb{Z}^\ell,\sigma_1}$,

$$\begin{aligned} \mathbf{c}_0^\star &= \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}_0^\star \in \mathbb{Z}_q^m \\ \mathbf{c}_1^\star &= \mathbf{R}^\top \cdot (\mathbf{c}_0^\star - \mathbf{e}_0^\star) + \mathbf{e}_1^\star + \boldsymbol{x}^\star \cdot \lfloor q/K \rfloor \in \mathbb{Z}_q^\ell \end{aligned}$$

Since the experiment knows the secret key, it is able to compute the ciphertext in this way. This change is only conceptual since the ciphertext is identical to that of Game 0.

Game 2: This game is the same as the previous one, except that the ciphertext is computed as: $\mathbf{e}_0^\star \hookleftarrow D_{\mathbb{Z}^m,\alpha q}$, $\mathbf{e}_1^\star \hookleftarrow D_{\mathbb{Z}^\ell,\sigma_1}$,

$$\begin{aligned} \mathbf{c}_0^\star &= \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}_0^\star \in \mathbb{Z}_q^m \\ \mathbf{c}_1^\star &= \mathbf{R}^\top \cdot \mathbf{c}_0^\star + \mathbf{e}_1^\star + \boldsymbol{x}^\star \cdot \lfloor q/K \rfloor \in \mathbb{Z}_q^\ell \end{aligned}$$

The only difference between the last two games is the term $\mathbf{R}^\top \cdot \mathbf{e}_0^\star$ in the expression of the ciphertext component $\mathbf{c}_1^\star$.

Since the norm of $\mathbf{R}^\top \cdot \mathbf{e}_0^\star$ is relatively small, the statistical discrepancy between the two games is "drowned", as the term $\mathbf{e}_1^\star$ is sampled from a very wide Gaussian distribution. Formally, by Lemma 2.9, Game 1 and Game 2 are statistically close, because the choice of parameters implies $\frac{\|\mathbf{R}^\top \cdot \mathbf{e}_0^\star\|_\infty}{\sigma_1} < 2^{-\lambda}$.

Game 3 This game is the same as the previous one, except that only the error $\mathbf{e}_1^\star \hookleftarrow D_{\mathbb{Z}^\ell,\sigma_1}$ is sampled and the ciphertext is computed as: ,

$$\begin{aligned} \mathbf{c}_0^\star &\hookleftarrow U(\mathbb{Z}_q^m) \\ \mathbf{c}_1^\star &= \mathbf{R}^\top \cdot \mathbf{c}_0^\star + \mathbf{e}_1^\star + \boldsymbol{x}^\star \cdot \lfloor q/K \rfloor \in \mathbb{Z}_q^\ell \end{aligned}$$

By the $\mathsf{LWE}_{q,m,n,\alpha}$ assumption the last two games are computationally indistinguishable.

Game 4: In this game, the experiment first runs $(\bar{\mathbf{A}}, \mathbf{T}_{\bar{\mathbf{A}}}) \leftarrow \mathsf{TrapGen}(1^m, 1^{n+1})$ and parses $\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} \\ \mathbf{a}_0^\top \end{bmatrix} \in \mathbb{Z}_q^{(n+1)\times m}$. Then, it samples $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m\times\ell},\sigma}$, computes $\mathbf{U} = \mathbf{AR}$ and sets $\mathsf{mpk}^\star = (\mathbf{A}, \mathbf{U})$ and $\mathsf{msk}^\star = (\mathbf{R}, \mathbf{T}_{\bar{\mathbf{A}}}, \mathbf{a}_0)$. Also samples $\mathbf{e}_1^\star \hookleftarrow D_{\mathbb{Z}^\ell,\sigma_1}$, and computes the ciphertext as:

$$\begin{aligned} \mathbf{c}_0^\star &= \mathbf{a}_0 \in \mathbb{Z}_q^m \\ \mathbf{c}_1^\star &= \mathbf{R}^\top \cdot \mathbf{c}_0^\star + \mathbf{e}_1^\star + \boldsymbol{x}^\star \cdot \lfloor q/K \rfloor \in \mathbb{Z}_q^\ell. \end{aligned} \tag{5.6}$$

The key queries are still answered as in the previous games. Notice that the difference between Game 3 and Game 4 is the way the matrix $\mathbf{A}$ and the vector $\mathbf{c}_0^\star$ are sampled. By Lemma 2.8, the distribution of the matrix $\begin{bmatrix}\mathbf{A}\\ \mathbf{a}_0^\top\end{bmatrix}$, which was generated using the TrapGen algorithm, is statistically close to the uniform distribution $U(\mathbb{Z}_q^{(n+1)\times m})$. This implies that Game 3 and Game 4 are statistically close in $\mathscr{A}$'s view.

Game 5: This game is exactly like the previous game, except that right after the generation of the keys, the experiment uses the trapdoor $\mathbf{T}_{\bar{\mathbf{A}}} \in \mathbb{Z}^{m\times m}$ to sample a "small" vector $\mathbf{w} \in \mathbb{Z}^m$ (as in Remark 1) such that

$$\begin{bmatrix}\mathbf{A}\\ \mathbf{a}_0^\top\end{bmatrix} \cdot \mathbf{w} = \begin{bmatrix}\mathbf{0}^n\\ \lfloor q/K \rfloor\end{bmatrix} \in \mathbb{Z}_q^{n+1},$$

then sets $\mathsf{msk}^\star = (\mathbf{R}, \mathbf{T}_{\bar{\mathbf{A}}}, \mathbf{a}_0, \mathbf{w})$

Notice that the last two games are the identical in the view of the adversary.

Game 6: This is identical to the Ideal experiment. In Lemma 5.7 we prove that Game 5 and Game 6 are statistically close, thus concluding the proof.

□

Lemma 5.7. *The advantage of any distinguisher between* Game_5 *and* Game_6 *is statistically negligible and* $\mathsf{Adv}^{56}_{\mathscr{A}}(\lambda) \le 2^{-\lambda}$.

Proof. In order to prove the claim, we simultaneously define Game'_5 and Game'_6 as follows. For each $k \in \{5,6\}$, define Game'_k identically to Game_k except that, at the outset of the game, the challenger samples $\Delta\boldsymbol{x} \hookleftarrow U([-\bar{X},\bar{X}]^\ell)$, where $\bar{X} = X + XY\cdot\ell^2\cdot(Y\sqrt{\ell})^\ell$. Before generating the challenge ciphertext, the challenger uses Lemma 2.15 to compute $\bar{\boldsymbol{x}} \in \mathbb{Z}^\ell$ such that $\|\bar{\boldsymbol{x}}\|_\infty \le \ell XY\cdot\ell(Y\sqrt{\ell})^\ell$ and $\mathbf{Y}_{\mathsf{pre}}\cdot\bar{\boldsymbol{x}} = \mathbf{Y}_{\mathsf{pre}}\cdot\boldsymbol{x}^\star$, where $\mathbf{Y}_{\mathsf{pre}}$ is the matrix obtained by stacking up the (linearly independent) transposed vectors $\boldsymbol{y}^\top$ occurring in pre-challenge queries. If $\Delta\boldsymbol{x} = \bar{\boldsymbol{x}} - \boldsymbol{x}^\star$ (we call this event Guess), the challenger proceeds as in Game_k. Otherwise, the challenger aborts the game and replaces $\mathscr{A}$'s output b' by a random bit.

As in the proof of Lemma 5.7, any adversary $\mathscr{A}$ that can distinguish between Game_5 and Game_6 with advantage $\mathsf{Adv}^{56}_{\mathscr{A}}(\lambda)$ can be used to distinguish between Game'_5 and Game'_6 with advantage

$$\mathsf{Adv}^{5'6'}_{\mathscr{A}}(\lambda) = \frac{1}{(2\bar{X})^\ell}\cdot\mathsf{Adv}^{56}_{\mathscr{A}}(\lambda). \tag{5.7}$$

Next, we show that $\mathsf{Adv}^{5'6'}_{\mathscr{A}}(\lambda) \le (2\bar{X})^{-\ell}\cdot 2^{-\lambda}$, which implies that Game_5 and Game_6 are statistically indistinguishable. To see this, observe that, when Guess occurs, Game'_6 is identical to a modification of Game'_5 where the master secret key has been replaced by

$$\mathbf{R}' = \mathbf{R} + \Delta\boldsymbol{x}^\top \otimes \mathbf{w} \in \mathbb{Z}^{m\times\ell}$$

, where $\mathbf{R} \hookleftarrow D_{\mathbb{Z}^{m\times\ell},\sigma}$ and $\Delta\boldsymbol{x}^\top \otimes \mathbf{w} = [\Delta x_1 \cdot \mathbf{w} | \cdots | \Delta x_\ell \cdot \mathbf{w}] \in \mathbb{Z}^{m\times\ell}$ is the Kronecker product matrix. Notice that the distribution of $\mathbf{R}'$ can be computed at the beginning of the experiment, thus we can define a game where $\mathbf{R}'$ is used instead of $\mathbf{R}$. By making this change of the master secret key in Game $5'$, all the pre-challenge keys remain invariant since

$$\mathbf{R}' \cdot \boldsymbol{y} = \mathbf{R}\boldsymbol{y} + (\Delta\boldsymbol{x}^\top \otimes \mathbf{w}) \cdot \boldsymbol{y} = \mathbf{R} \cdot \boldsymbol{y}, \text{ as long as } \Delta\boldsymbol{x}^\top \cdot \boldsymbol{y} = 0,$$

and also the public key remains invariant, because the way we chose $\mathbf{w} \in \mathbb{Z}^m$. Indeed, since $\mathbf{A} \cdot \mathbf{w} = \mathbf{0}^n \bmod q$ we have that $\mathbf{A} \cdot \mathbf{R}' = \mathbf{A} \cdot \mathbf{R} \bmod q$.

Hence the statistical distance between these two games is bounded by the statistical distance $\Delta\left(\mathbf{R}, \mathbf{R}'\right)$. We use the noise flood Lemma 2.9 to obtain the bound:

$$\Delta(\mathbf{R}, \mathbf{R}') \leq m\ell \cdot \frac{\|\Delta\boldsymbol{x}\|_\infty \cdot \|\mathbf{w}\|_\infty}{\sigma} < 2^{-\lambda}.$$

□

Chapter

6

Simulation-Sound NIZK Arguments for LWE-based Encryption

In this chapter we discuss the details of the generic transformations presented in [LNPT19], that yield non-interactive zero-knowledge (NIZK) argument systems. The first result is a generic compiler that gives *multi-theorem* NIZK arguments directly from trapdoor Σ-protocols, assuming lossy encryption and correlation-intractable hash functions.

Our second result improves the generic compiler, such that it directly transforms trapdoor Σ-protocols into *unbounded simulation-sound* NIZKs. Again, we need to assume some form of lossy encryption, correlation-intractable hash functions but also one-time signatures. The observation that some forms of lossy encryption admit an efficient opening is crucially used by the NIZK simulators in the above results. An efficient opening is an algorithm that takes input a lossy ciphertext and a message and outputs some randomness that explains the ciphertext as the encryption of this particular message.

It is also possible to obtain unbounded simulation sound NIZK by combining the generic NIZK techniques from [FLS99, SCO+01] and the latest results on instantiating the Fiat-Shamir paradigm without random oracles [CLW19, PS19]. But the resulting scheme would be terrible inefficient as it would have to go through a Karp reduction to the graph Hamiltonicity language. Our results enable more efficient NIZK constructions for languages of interest, because we can bypass the Karp reduction.

We can apply these ideas to obtain the most efficient public-key scheme, key-dependent message (KDM) [BRS02] secure against adaptive chosen-ciphertext attacks (CCA2), under the standard LWE assumption. KDM security guarantees privacy even when the adversary gets encryptions of messages that depend of the secret key itself. Constructions of public-key KDM-CCA2 secure schemes under

lattice assumptions, were previously known to be possible from any KDM-CPA scheme, by combining generic NIZK techniques [FLS99, SCO+01] and the recent results of [CLW19, PS19] with the Naor-Yung [NY90] transform. Efficient KDM-CCA constructions from standard DDH or DCR assumptions are known as well [HLL16, KT18, KMT19], but not from lattice assumptions.

To construct more efficient KDM-CCA2 secure schemes under standard lattice assumptions, we give a construction for a trapdoor Σ-protocol, that proves two ciphertexts of the LWE-based KDM-CPA secure scheme of [ACPS09] encrypt the same message. Together with the compiler for unbounded simulation-soundness, it allows us to apply the Naor-Yung transform to the ACPS scheme. This yields the most efficient public-key scheme, key-dependent message (KDM) secure under chosen-ciphertext attacks (CCA2), under the standard LWE assumption.

Organization

We start the last part of this thesis, with Section 6.1, by defining all the building-blocks that we use in this chapter, together with their security properties. Next, in Section 6.2 we give the construction and the security proof for the R_{BM}-lossy public-key encryption scheme with efficient opening, that we need for the unbounded simulation-sound NIZK transformation of Section 6.4. In Section 6.3 we present the generic construction of multi-theorem NIZKs and discuss an application of this result in Section 6.3.4.

6.1 Definitions

In this section we present the notions that we need for our results.

6.1.1 Trapdoor Sigma Protocols

Canetti *et al.* [CLW19] considered a definition of Σ-protocols that slightly differs from the usual formulation [CDS94, Cra96].

Definition 6.1 (Adapted from [CLW19, AJW12])**.** *Let a language* $\mathscr{L} = (\mathscr{L}_{\mathsf{zk}}, \mathscr{L}_{\mathsf{sound}})$ *associated with two NP relations* $R_{\mathsf{zk}}, R_{\mathsf{sound}}$*. A* 3*-move interactive proof system* $\Pi = (\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_{\mathscr{L}}, \mathsf{P}, \mathsf{V})$ *in the common reference string model is a Gap* Σ*-protocol for* $\mathscr{L}$ *if it satisfies the following conditions:*

- ***3-Move Form:*** *The prover and the verifier both take as input* $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathscr{L}})$*, with* $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$ *and* $\mathsf{crs}_{\mathscr{L}} \leftarrow \mathsf{Gen}_{\mathscr{L}}(\mathsf{par}, \mathscr{L})$*, and a statement* x *and proceed as follows: (i)* P *takes in* $w \in R_{\mathsf{zk}}(x)$*, computes* $(\mathbf{a}, st) \leftarrow \mathsf{P}(\mathsf{crs}, x, w)$ *and sends* $\mathbf{a}$ *to the verifier; (ii)* V *sends back a random challenge* Chall *from the challenge space* ChSp*; (iii)* P *finally sends a response* $\mathbf{z} = \mathsf{P}(\mathsf{crs}, x, w, \mathbf{a}, \mathsf{Chall}, st)$ *to* V*; (iv) On input of* $(\mathbf{a}, \mathsf{Chall}, \mathbf{z})$*,* V *outputs* 1 *or* 0*.*

- ***Completeness:*** *If* $(x, w) \in R_{\mathsf{zk}}$ *and* P *honestly computes* $(\mathbf{a}, \mathbf{z})$ *for a challenge* Chall*,* $\mathsf{V}(\mathsf{crs}, x, (\mathbf{a}, \mathsf{Chall}, \mathbf{z}))$ *outputs* 1 *with probability* $1 - \mathsf{negl}(\lambda)$*.*

- ***Special zero-knowledge:*** *There is a PPT simulator* ZKSim *that, on input of* crs*,* $x \in \mathscr{L}_{\mathsf{zk}}$ *and a challenge* $\mathsf{Chall} \in \mathsf{ChSp}$*, outputs* $(\mathbf{a}, \mathbf{z}) \leftarrow \mathsf{ZKSim}(\mathsf{crs}, x, \mathsf{Chall})$ *such that* $(\mathbf{a}, \mathsf{Chall}, \mathbf{z})$ *is computationally indistinguishable from a real transcript with challenge* Chall *(for* $w \in R_{zk}(x)$*).*

- ***Special soundness:*** *For any* $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathscr{L}})$ *obtained as* $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$*,* $\mathsf{crs}_{\mathscr{L}} \leftarrow \mathsf{Gen}_{\mathscr{L}}(\mathsf{par}, \mathscr{L})$*, any* $x \notin \mathscr{L}_{\mathsf{sound}}$*, and any first message* $\mathbf{a}$ *sent by* P*, there is at most one challenge* $\mathsf{Chall} = f(\mathsf{crs}, x, \mathbf{a})$ *for which an accepting transcript* $(\mathsf{crs}, x, \mathbf{a}, \mathsf{Chall}, \mathbf{z})$ *exists for some third message* $\mathbf{z}$*. The function* f *is called the "bad challenge function" of* Π*. That is, if* $x \notin \mathscr{L}_{\mathsf{sound}}$ *and the challenge differs from the bad challenge, the verifier never accepts.*

Definition 6.1 is taken from [CLW19, AJW12] and relaxes the standard special soundness property in that extractability is not required. Instead, it considers a bad challenge function f, which may not be efficiently computable. Canetti *et al.* [CLW19] define *trapdoor* Σ-protocols as Σ-protocols where the bad challenge function is efficiently computable using a trapdoor. They also define instance-dependent trapdoor Σ-protocol where the trapdoor τ_Σ should be generated as a function of some instance $x \notin \mathcal{L}_{\mathsf{sound}}$. Here, we use a definition where x need not be known in advance (which is not possible for instance in applications to chosen-ciphertext security, where x is determined by a decryption query) and the trapdoor does not depend on a specific x. However, the common reference string and the trapdoor may depend on the language.

The common reference string $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathcal{L}})$ consists of a fixed part par and a language-dependent part $\mathsf{crs}_{\mathcal{L}}$ which is generated as a function of par and a language parameter $\mathcal{L} = (\mathcal{L}_{\mathsf{zk}}, \mathcal{L}_{\mathsf{sound}})$.

Definition 6.2 (Adapted from [CLW19])**.** *A Σ-protocol $\Pi = (\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_{\mathcal{L}}, \mathsf{P}, \mathsf{V})$ with bad challenge function f for a trapdoor language $\mathcal{L} = (\mathcal{L}_{\mathsf{zk}}, \mathcal{L}_{\mathsf{sound}})$ is a* ***trapdoor Σ-protocol*** *if it satisfies the properties of Definition 6.1 and there exist PPT algorithms* $(\mathsf{TrapGen}, \mathsf{BadChallenge})$ *with the following properties.*

- $\mathsf{Gen}_{\mathsf{par}}$ *inputs $\lambda \in \mathbb{N}$ and outputs public parameters* $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$.
- $\mathsf{Gen}_{\mathcal{L}}$ *is a randomized algorithm that, on input of public parameters* par*, outputs the language-dependent* $\mathsf{crs}_{\mathcal{L}} \leftarrow \mathsf{Gen}_{\mathcal{L}}(\mathsf{par}, \mathcal{L})$ *of* $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathcal{L}})$.
- $\mathsf{TrapGen}(\mathsf{par}, \mathcal{L}, \tau_{\mathcal{L}})$ *takes as input public parameters* par *and a membership-testing trapdoor $\tau_{\mathcal{L}}$ for the language $\mathcal{L}_{\mathsf{sound}}$. It outputs a common reference string* $\mathsf{crs}_{\mathcal{L}}$ *and a trapdoor τ_Σ.*
- $\mathsf{BadChallenge}(\tau_\Sigma, \mathsf{crs}, x, \mathbf{a})$ *takes in a trapdoor τ_Σ, a CRS* $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathcal{L}})$*, an instance x, and a first prover message* $\mathbf{a}$*. It outputs a challenge* Chall.

In addition, the following properties are required.

- ***CRS indistinguishability:*** *For any* $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$*, and any trapdoor $\tau_{\mathcal{L}}$ for the language $\mathcal{L}$, an honestly generated* $\mathsf{crs}_{\mathcal{L}}$ *is computationally indistinguishable from a CRS produced by* $\mathsf{TrapGen}(\mathsf{par}, \mathcal{L}, \tau_{\mathcal{L}})$*. Namely, for any PPT distinguisher $\mathcal{A}$, we have*

$$\begin{aligned}\mathbf{Adv}^{\text{indist-}\Sigma}_{\mathcal{A}}(\lambda) := |\Pr[\mathsf{crs}_{\mathcal{L}} \leftarrow \mathsf{Gen}_{\mathcal{L}}(\mathsf{par}, \mathcal{L}) : \mathcal{A}(\mathsf{par}, \mathsf{crs}_{\mathcal{L}}) = 1] \\ - \Pr[(\mathsf{crs}_{\mathcal{L}}, \tau_\Sigma) \leftarrow \mathsf{TrapGen}(\mathsf{par}, \mathcal{L}, \tau_{\mathcal{L}}) : \mathcal{A}(\mathsf{par}, \mathsf{crs}_{\mathcal{L}}) = 1]| \leq \mathsf{negl}(\lambda).\end{aligned}$$

- ***Correctness:*** *For all pairs* $(\mathsf{crs}_{\mathcal{L}}, \tau_\Sigma) \leftarrow \mathsf{TrapGen}(\mathsf{par}, \mathcal{L}, \tau_{\mathcal{L}})$*, where $\tau_{\mathcal{L}}$ is a language-specific trapdoor, and for any instance $x \notin \mathcal{L}_{\mathsf{sound}}$ we have*

$$\mathsf{BadChallenge}(\tau_\Sigma, \mathsf{crs}, x, \mathbf{a}) = f(\mathsf{crs}, x, \mathbf{a}) \ .$$

Note that the TrapGen algorithm does not take a specific statement x as input, but only a trapdoor $\tau_{\mathcal{L}}$ allowing to recognize elements of $\mathcal{L}_{\mathsf{sound}}$.

6.1.2 NIZKs and Simulation-Sound Proofs

We recall the definitions of NIZK proofs. Since it is sufficient for our applications, we allow the common reference string to be generated as a function of the language $\mathscr{L}_R$.

Definition 6.3. *A non-interactive zero-knowledge (NIZK) argument system* Π *for a class of NP relations* $\mathscr{R}$ *consists of four PPT algorithms* $(\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_{\mathscr{L}}, \mathsf{P}, \mathsf{V})$ *with the following syntax:*

- $\mathsf{Gen}_{\mathsf{par}}(1^\lambda)$ *inputs a security parameter* λ *and outputs public parameters* par.
- $\mathsf{Gen}_{\mathscr{L}}(1^\lambda, R)$ *takes as input a security parameter* λ *and the description of a relation* $R \in \mathscr{R}$ *which specifies a statement length* N. *It outputs the language-dependent part* $\mathsf{crs}_{\mathscr{L}}$ *of the common reference string* $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathscr{L}})$.
- $\mathsf{P}(\mathsf{crs}, x, w)$ *is a proving algorithm taking as input the common reference string* crs, *a statement* $x \in \{0,1\}^N$ *and a witness* w *such that* $(x, w) \in R$. *It outputs a proof* π.
- $\mathsf{V}(\mathsf{crs}, x, \pi)$ *is a verification algorithm taking as input a common reference string* crs, *a statement* $x \in \{0,1\}^N$, *and a proof* π. *It outputs* 1 *or* 0.

Moreover, Π *should satisfy the following properties. For simplification we denote below by* Setup *an algorithm that runs successively* $\mathsf{Gen}_{\mathsf{par}}$ *and* $\mathsf{Gen}_{\mathscr{L}}$ *to generate a common reference string.*

- ***Completeness:*** *For any* $(x, w) \in R$, *we have*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, R),\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

- ***Soundness (non-adaptive):*** *Let the language* $\mathscr{L}_R = \{x \mid \exists w \ \ s.t. \ \ (x, w) \in R\}$ *associated with* R. *For any* $x \in \{0,1\}^N \setminus \mathscr{L}_R$ *and any PPT prover* P^*, *we have*

$$\Pr\left[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, R),\ \pi \leftarrow P^*(\mathsf{crs}, x) : \mathsf{V}(\mathsf{crs}, x, \pi) = 1\right] \leq \mathsf{negl}(\lambda) \ .$$

- ***Zero-Knowledge:*** *There is a PPT simulator* $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that, for any PPT adversary* $\mathscr{A}$, *we have*

$$\begin{aligned} &|\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, R) \ : \ \mathscr{A}^{\mathsf{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) = 1] \\ &\qquad - \Pr[(\mathsf{crs}, \tau_{\mathsf{zk}}) \leftarrow \mathsf{Sim}_0(1^\lambda, R) \ : \ \mathscr{A}^{\mathscr{O}(\mathsf{crs},\tau_{\mathsf{zk}},\cdot,\cdot)}(\mathsf{crs}) = 1]| \leq \mathsf{negl}(\lambda) \ . \end{aligned}$$

  Here, $\mathsf{P}(\mathsf{crs}, \cdot, \cdot)$ *is an oracle that outputs* $\perp$ *on input of* $(x, w) \notin R$ *and outputs a valid proof* $\pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)$ *otherwise;* $\mathscr{O}(\mathsf{crs}, \tau_{\mathsf{zk}}, \cdot, \cdot)$ *is an oracle that outputs* $\perp$ *on input of* $(x, w) \notin R$ *and outputs a simulated proof* $\pi \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathsf{zk}}, x)$ *on input of a pair* $(x, w) \in R$. *Note that this simulated proof* π *is generated independently of the witness* w *provided as input.*[1]

[1] In particular, Sim_1 can be run on any statement x, even $x \notin \mathscr{L}_R$. This is central in the definition of unbounded simulation soundness (Definition 6.4).

Definition 6.3 captures a notion of multi-theorem zero-knowledge, which allows the adversary to obtain proofs for multiple statements. Feige *et al.* [FLS99] gave a generic transformation of a multi-theorem NIZK argument system from a single-theorem one (where the adversary can only invoke the oracle once).

SIMULATION-SOUNDNESS. We now recall the definition of simulation-soundness introduced in [Sah99], which informally captures the adversary's inability to create a new proof for a false statement $x^\star$ even after having seen simulated proofs for possibly false statements $\{x_i\}_i$ of its choice.

In the following, in order to allow a challenger to efficiently check the winning condition (ii) in the security experiment, we restrict ourselves to *trapdoor languages*, where a language-specific trapdoor $\tau_{\mathscr{L}}$ makes it possible to determine if a given statement $x^\star \in \{0,1\}^N$ belongs to the language $\mathscr{L}_R$ with overwhelming probability.

Definition 6.4 ([Sah99, SCO+01])**.** *A NIZK argument system for a class of relations $\mathscr{R}$ provides* ***unbounded simulation soundness*** *if no PPT adversary has noticeable advantage in this game.*

1. *The challenger chooses a relation $R \in \mathscr{R}$ together with a membership testing trapdoor $\tau_{\mathscr{L}}$ that allows recognizing elements of $\mathscr{L}_R$. Let $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ be an efficient NIZK simulator for $\mathscr{L}_R$. The challenger generates $(\mathsf{crs}, \tau_{\mathrm{zk}}) \leftarrow \mathsf{Sim}_0(1^\lambda, R)$ and gives $(\mathsf{crs}, \tau_{\mathscr{L}})$ to the adversary $\mathscr{A}$.*

2. *$\mathscr{A}$ is given oracle access to $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot)$. At each query, $\mathscr{A}$ chooses a statement $x \in \{0,1\}^N$ and obtains $\pi \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, x)$.*

3. *$\mathscr{A}$ outputs $(x^\star, \pi^\star)$.*

Let $\mathscr{Q}$ be the set of all simulation queries and responses (x_i, π_i) made by $\mathscr{A}$. The adversary $\mathscr{A}$ wins if the following conditions are satisfied: (i) $(x^\star, \pi^\star) \notin \mathscr{Q}$; (ii) $x^\star \notin \mathscr{L}_R$; and (iii) $\mathsf{V}(\mathsf{crs}, x^\star, \pi^\star) = 1$. The adversary's advantage $\mathbf{Adv}^{\mathrm{uss}}_{\mathscr{A}}(\lambda)$ is its probability of success taken over all coin tosses.

6.1.3 Correlation Intractable Hash Functions

We consider unique-output searchable binary relations [CCH+19]. These are binary relations such that, for every x, there is at most one y such that $R(x,y) = 1$ and y is efficiently computable from x.

Definition 6.5. *A relation $R \subseteq \mathscr{X} \times \mathscr{Y}$ is* ***searchable*** *in size S if there exists a function $f : \mathscr{X} \to \mathscr{Y}$ which is computable by a size S boolean circuit such that, if there exists y such that $(x,y) \in R$, then $f(x) = y$.*

Letting $\lambda \in \mathbb{N}$ denote a security parameter, a hash family with input length $n(\lambda)$ and output length $m(\lambda)$ is a collection $\mathscr{H} = \{h_\lambda : \{0,1\}^{s(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}$

of keyed hash functions implemented by efficient algorithms $(\mathsf{Gen}, \mathsf{Hash})$, where $\mathsf{Gen}(1^\lambda)$ outputs a key $k \in \{0,1\}^{s(\lambda)}$ and $\mathsf{Hash}(k,x)$ computes a hash value $h_\lambda(k,x) \in \{0,1\}^{m(\lambda)}$.

Definition 6.6. *For a relation class $\mathscr{R} = \{\mathscr{R}_\lambda \subseteq \{0,1\}^{n(\lambda)} \times \{0,1\}^{m(\lambda)}\}$, i.e. a set of relations for each λ, a hash function family $\mathscr{H} = \{h_\lambda : \{0,1\}^{s(\lambda)} \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\}$ is **$\mathscr{R}$-correlation intractable** if, for any probabilistic polynomial time (PPT) adversary $\mathscr{A}$ there is a negligible function ν, such that for any $R \in \mathscr{R}_\lambda$ we have*

$$\Pr\left[k \leftarrow \mathsf{Gen}(1^\lambda),\ x \leftarrow \mathscr{A}(k) : (x, h_\lambda(k,x)) \in R\right] \leq \nu(\lambda).$$

Definition 6.7 ([CLW19, CCH+19])**.** *Given a relation ensemble $\mathscr{R} = \{\mathscr{R}_\lambda\}$, a hash family $\mathscr{H}$ is **somewhere statistically correlation intractable** w.r.t. $\mathscr{R}$ if there is an efficient algorithm* StatGen *with the following properties:*

- $\mathsf{StatGen}(1^\lambda, \mathsf{aux})$ *is a fake key generation that takes as input a security parameter λ and an auxiliary input* aux*. It outputs a hashing key k.*

- *For any relation $R \in \mathscr{R}_\lambda$, there is an auxiliary input aux_R with the following properties:*

  - ***Key indistinguishability:** The distributions $\{k \mid k \leftarrow \mathsf{Gen}(1^\lambda)\}$ and $\{k \mid k \leftarrow \mathsf{StatGen}(1^\lambda, \mathsf{aux}_R)\}$ are computationally indistinguishable. For any PPT distinguisher $\mathscr{A}$, the following function should be negligible:*

$$\mathbf{Adv}^{\text{indist-CI}}_{\mathscr{A}}(\lambda) := |\Pr[k \leftarrow \mathsf{Gen}(1^\lambda) : \mathscr{A}(k, \mathsf{aux}_R) = 1] - \Pr[k \leftarrow \mathsf{StatGen}(1^\lambda, \mathsf{aux}_R) : \mathscr{A}(k, \mathsf{aux}_R) = 1]| \ .$$

  - ***Statistical Correlation Intractability:** With overwhelming probability over the choice of $k \leftarrow \mathsf{StatGen}(1^\lambda, \mathsf{aux}_R)$, no pair $(k, h(k,x))$ satisfies R:*

$$\Pr_{k \leftarrow \mathsf{StatGen}(1^\lambda, \mathsf{aux}_R)}\left[\exists x \in \{0,1\}^{n(\lambda)} \ :\ (x, h(k,x)) \in R\right] \leq 2^{-\Omega(\lambda)} \ .$$

Peikert and Shiehian [PS19] described a somewhere correlation-intractable hash family for any searchable relation (in the sense of Definition 6.5) defined by functions f of bounded depth. Their construction relies on the standard LWE assumption with polynomial approximation factors.

Here, we explicitly require the key indistinguishability property to hold even when the adversary is given the auxiliary information aux_R associated with the relation. However, this property is satisfied by the LWE-based construction of [PS19], where aux_R is the description of the circuit that evaluates the relation.

6.1.4 Strongly Unforgeable One-Time-Signatures

Definition 6.8. *A signature scheme* $\mathsf{OTS} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ *is Strongly Unforgeable One-Time-Signature (OTS) if any* PPT *adversary* $\mathcal{A}$ *has only negligible advantage in producing a forgery in the following game.*

1. $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathcal{G}(1^\lambda)$ *and the verification key* VK *is given to* $\mathcal{A}$.
2. *The adversary can obtain a signature* $\sigma \leftarrow \mathcal{S}(\mathsf{SK}, m)$ *for a single message* m *of his choosing.*
3. *The adversary outputs a forgery* $(m^\star, \sigma^\star)$.

We define the advantage of $\mathcal{A}$ *winning the game as:*

$$\mathbf{Adv}^{\mathrm{ots}}_{\mathcal{A}}(1^\lambda) := \Pr\left[\mathcal{V}(\mathsf{VK}, (m^\star, \sigma^\star)) = 1 \wedge (m^\star, \sigma^\star) \neq (m, \sigma)\right],$$

where the probabilities are taken over $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathcal{G}(1^\lambda)$, $(m^\star, \sigma^\star) \leftarrow \mathcal{A}(\mathsf{VK}, \sigma)$, $\sigma \leftarrow \mathcal{S}(\mathsf{SK}, m)$ *and the internal randomness of* $\mathcal{A}$.

6.1.5 Lossy Encryption With Efficient Opening

We recall the notion of lossy encryption with efficient opening as considered by Bellare *et al.* [BHY09].

Definition 6.9. *A lossy PKE scheme with efficient opening is a tuple of PPT algorithms* (Par-Gen, Keygen, LKeygen, Encrypt, Decrypt, Opener) *such that:*

Public parameters: Par-Gen *inputs a security parameter* $\lambda \in \mathbb{N}$ *and outputs public parameters* Γ, *which specify a message space* MsgSp, *a ciphertext space* CtSp *and a randomness space* R^{LPKE}.

Key generation: *On input of public parameters* Γ, Keygen *outputs an injective public key* $pk \in \mathcal{PK}$ *and a secret key* $sk \in \mathcal{SK}$.

Lossy Key generation: *On input of public parameters* Γ, LKeygen *outputs a lossy public key* $pk \in \mathcal{PK}$ *and a lossy secret key* $sk \in \mathcal{SK}$.

Decryption under injective keys: *For any* $\Gamma \leftarrow \mathsf{Par\text{-}Gen}(1^\lambda)$, *any injective key pair* $(pk, sk) \leftarrow \mathsf{Keygen}(\Gamma)$, *and any message* $\mathsf{Msg} \in \mathsf{MsgSp}$, *we have*

$$\Pr\left[\exists r \in R^{\mathsf{LPKE}} : \mathsf{Decrypt}\big(sk, \mathsf{Encrypt}(pk, \mathsf{Msg}; r)\big) \neq \mathsf{Msg}\right] < \nu(\lambda) \ ,$$

for some negligible function $\nu(\lambda)$, *where* $(pk, sk) \leftarrow \mathsf{Keygen}(1^\lambda)$ *and the probability is taken over the randomness of* Keygen.

Indistinguishability: *The distributions* $D_{\text{inj}} = \{pk \mid (pk, sk) \leftarrow \mathsf{Keygen}(1^\lambda)\}$ *and* $D_{\text{loss}} = \{pk \mid (pk, sk) \leftarrow \mathsf{LKeygen}(1^\lambda)\}$ *are indistinguishable. For any PPT adversary, we have* $\mathbf{Adv}^{\mathsf{indist\text{-}LPKE}}(\lambda) \leq \mathsf{negl}(\lambda)$*, where*

$$\mathbf{Adv}^{\mathsf{indist\text{-}LPKE}}(\lambda) := |\Pr[pk \hookleftarrow D_{\text{inj}} : \mathscr{A}(pk) = 1] - \Pr[pk \hookleftarrow D_{\text{loss}} : \mathscr{A}(pk) = 1]| \ .$$

Lossiness under lossy keys: *For any* $(pk, sk) \leftarrow \mathsf{LKeygen}(1^\lambda)$ *and any two messages* $\mathsf{Msg}_0, \mathsf{Msg}_1 \in \mathsf{MsgSp}$*, the distributions*

$$\{C \mid C \leftarrow \mathsf{Encrypt}(pk, \mathsf{Msg}_0)\} \approx_s \{C \mid C \leftarrow \mathsf{Encrypt}(pk, \mathsf{Msg}_1)\},$$

are statistically close.

Efficient opening under lossy keys: *Let* D_R *the distribution over* R^{LPKE}*, that samples random coins for the* Encrypt *algorithm. For any* $\mathsf{Msg} \in \mathsf{MsgSp}$ *and ciphertext* C*, let* $D_{PK,\mathsf{Msg},C}$ *denote the probability distribution on* R^{LPKE} *with support* $S_{PK,\mathsf{Msg},C} = \{\overline{r} \in R^{\mathsf{LPKE}} \mid \mathsf{Encrypt}(pk, \mathsf{Msg}, \overline{r}) = C\}$*, and such that, for each* $\overline{r} \in S_{PK,\mathsf{Msg},C}$*, we have*

$$D_{PK,\mathsf{Msg},C}(\overline{r}) = \Pr_{r' \hookleftarrow D_R}[r' = \overline{r} \mid \mathsf{Encrypt}(pk, \mathsf{Msg}, r') = C] \ .$$

There is a PPT sampling algorithm Opener *such that, for any keys* $(pk, sk) \leftarrow \mathsf{LKeygen}(1^\lambda)$*, any randomness* $r \hookleftarrow D_R$*, and any* $\mathsf{Msg}_0, \mathsf{Msg}_1 \in \mathsf{MsgSp}$*, takes as inputs* $C = \mathsf{Encrypt}(pk, \mathsf{Msg}_0, r)$*,* Msg_1 *and the lossy keys* (pk, sk) *and outputs an independent sample* $\overline{r}$ *from a distribution that is statistically close to* $D_{PK,\mathsf{Msg}_1,C}$*.*

The variant of Regev's cryptosystem suggested in [GPV08, Section 8.1] where encryption coins are sampled from a Gaussian distribution is a Lossy PKE with efficient opening.

6.1.6 R-Lossy Public-Key Encryption With Efficient Opening

We generalize the lossy encryption notion from the previous section and the notion of R-lossy public-key encryption introduced by Boyle *et al.* [BSW11a]. As defined in [BSW11a], it is a tag-based encryption scheme [Kil06] where the tag space $\mathscr{T}$ is partitioned into a set of *injective* tags and a set of *lossy* tags. When ciphertexts are generated for an injective tag, the decryption algorithm correctly recovers the underlying plaintext. When messages are encrypted under lossy tags, the ciphertext is statistically independent of the plaintext. In R-lossy PKE schemes, the tag space is partitioned according to a binary relation $R \subseteq \mathscr{K} \times \mathscr{T}$. The key generation algorithm takes as input an initialization value $K \in \mathscr{K}$ and partitions $\mathscr{T}$ in such a way that injective tags $t \in \mathscr{T}$ are exactly those for which $(K, t) \in R$ (i.e., all tags t for which $(K, t) \notin R$ are lossy).

From a security standpoint, we require the existence of a lossy key generation algorithm $\mathsf{LKeygen}$ which outputs public keys with respect to which all tags t are lossy (in contrast with injective keys where the only lossy tags are those for which $(K,t) \notin R$). Second, we also ask that the secret key makes it possible to equivocate lossy ciphertexts (a property called *efficient opening* by Bellare *et al.* [BHY09]) using an algorithm called Opener. Finally, we use two distinct opening algorithms Opener and $\mathsf{LOpener}$. The former operates over injective public keys for lossy tags while the latter can equivocate ciphertexts encrypted under lossy keys for any tag.

Definition 6.10. *Let $R \subseteq \mathcal{K}_\lambda \times \mathcal{T}_\lambda$ be an efficiently computable binary relation. An R-lossy PKE scheme with efficient opening is a 7-uple of PPT algorithms,*
$(\mathsf{Par\text{-}Gen}, \mathsf{Keygen}, \mathsf{LKeygen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Opener}, \mathsf{LOpener})$*, such that:*

Parameter generation: *On input a security parameter λ, $\mathsf{Par\text{-}Gen}(1^\lambda)$ outputs public parameters Γ.*

Key generation: *For an initialization value $K \in \mathcal{K}_\lambda$ and public parameters Γ, algorithm $\mathsf{Keygen}(\Gamma, K)$ outputs an injective public key $pk \in \mathcal{PK}$, a decryption key $sk \in \mathcal{SK}$ and a trapdoor key $tk \in \mathcal{TK}$. The public key specifies a a message space MsgSp, ciphertext space CtSp and a randomness space R^{LPKE}.*

Lossy Key generation: *Given an initialization value $K \in \mathcal{K}_\lambda$ and public parameters Γ, the lossy key generation algorithm $\mathsf{LKeygen}(\Gamma, K)$ outputs a lossy public key $pk \in \mathcal{PK}$, a lossy secret key $sk \in \mathcal{SK}$ and a trapdoor key $tk \in \mathcal{TK}$.*

Decryption under injective tags: *For any initialization value $K \in \mathcal{K}$, any tag $t \in \mathcal{T}$ such that $(K,t) \in R$, and any message $\mathsf{Msg} \in \mathsf{MsgSp}$, we have*

$$\Pr\left[\exists r \in R^{\mathsf{LPKE}} : \mathsf{Decrypt}\big(sk, t, \mathsf{Encrypt}(pk, t, \mathsf{Msg}; r)\big) \neq \mathsf{Msg}\right] < \nu(\lambda) \ ,$$

for some negligible function $\nu(\lambda)$, where $(pk, sk, tk) \leftarrow \mathsf{Keygen}(\Gamma, K)$ and the probability is taken over the randomness of Keygen.

Indistinguishability: *Algorithms* $\mathsf{LKeygen}$ *and* Keygen *satisfy the following:*

(i) For any $K \in \mathcal{K}_\lambda$, the distributions

$$D_{\text{inj}} = \{(pk, tk) \mid (pk, sk, tk) \leftarrow \mathsf{Keygen}(\Gamma, K)\},$$

$$D_{\text{loss}} = \{(pk, tk) \mid (pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K)\}$$

are computationally indistinguishable. For any PPT adversary $\mathcal{A}$, we have $\mathbf{Adv}^{\mathsf{indist\text{-}LPKE\text{-}1}}_{\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$, where

$$\mathbf{Adv}^{\mathsf{indist\text{-}LPKE\text{-}1}}_{\mathcal{A}}(\lambda) := |\Pr[(pk, tk) \leftarrow\!\!\rightarrow D_{\text{inj}} : \mathcal{A}(pk, tk) = 1] - \Pr[(pk, tk) \leftarrow\!\!\rightarrow D_{\text{loss}} : \mathcal{A}(pk, tk) = 1]| \ .$$

(ii) For any distinct initialization values $K, K' \in \mathcal{K}_\lambda$, the following distributions

$$\{pk \mid (pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K)\},$$

$$\{pk \mid (pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K')\}$$

are indistinguishable. The advantage of any PPT distinguisher $\mathcal{A}$ is denoted by $\mathbf{Adv}^{\text{indist-LPKE-2}}_{\mathcal{A}}(\lambda)$ and defined as usual.

Lossiness under lossy tags: *For any initialization value $K \in \mathcal{K}_\lambda$ and tag $t \in \mathcal{T}_\lambda$ such that $(K, t) \notin R$, any $(pk, sk, tk) \leftarrow \mathsf{Keygen}(\Gamma, K)$, and any $\mathsf{Msg}_0, \mathsf{Msg}_1 \in \mathsf{MsgSp}$, the following distributions are statistically close:*

$$\{C \mid C \leftarrow \mathsf{Encrypt}(pk, t, \mathsf{Msg}_0)\} \quad \approx_s \quad \{C \mid C \leftarrow \mathsf{Encrypt}(pk, t, \mathsf{Msg}_1)\}.$$

Efficient opening under lossy tags: *Let D_R denote the distribution, defined over the randomness space R^{LPKE}, from which the random coins used by* Encrypt *are sampled. For any message $\mathsf{Msg} \in \mathsf{MsgSp}$ and ciphertext C, let $D_{PK,\mathsf{Msg},C,t}$ denote the probability distribution on R^{LPKE} with support*

$$S_{PK,\mathsf{Msg},C,t} = \{\overline{r} \in R^{\mathsf{LPKE}} \mid \mathsf{Encrypt}(pk, t, \mathsf{Msg}, \overline{r}) = C\} \ ,$$

and such that, for each $\overline{r} \in S_{PK,\mathsf{Msg},C,t}$, we have

$$D_{PK,\mathsf{Msg},C,t}(\overline{r}) = \Pr_{r' \hookleftarrow D_R}[r' = \overline{r} \mid \mathsf{Encrypt}(pk, t, \mathsf{Msg}, r') = C] \ .$$

There exists a PPT algorithm Opener *such that, for any $K \in \mathcal{K}_\lambda$, any keys $(pk, sk, tk) \leftarrow \mathsf{Keygen}(\Gamma, K)$ and $(pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K)$, any random coins $r \hookleftarrow D_R$, any tag $t \in \mathcal{T}_\lambda$ such that $(K, t) \notin R$, and any $\mathsf{Msg}_0, \mathsf{Msg}_1 \in \mathsf{MsgSp}$, takes as inputs pk, Msg_1, $C = \mathsf{Encrypt}(pk, t, \mathsf{Msg}_0, r)$, t and tk. It outputs a sample $\overline{r}$ from a distribution statistically close to $D_{PK,\mathsf{Msg}_1,C,t}$.*

Efficient opening under lossy keys: *There exists an efficient sampling algorithm* LOpener *such that, for any $K \in \mathcal{K}_\lambda$, any keys $(pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K)$, any random coins $r \hookleftarrow D_R$, any tag $t \in \mathcal{T}_\lambda$, and any distinct $\mathsf{Msg}_0, \mathsf{Msg}_1 \in \mathsf{MsgSp}$, takes as input pk, Msg_1, $C = \mathsf{Encrypt}(pk, t, \mathsf{Msg}_0, r)$, t and sk. It outputs a sample $\overline{r}$ from a distribution statistically close to $D_{PK,\mathsf{Msg}_1,C,t}$.*

In Definition 6.10, some of the first four properties were defined in [BSW11a, Definition 4.1]. The last two properties are a natural extension of the definition of efficient opening introduced by Bellare *et al.* [BHY09]. We note that property of decryption under injective tags does not assume that random coins are honestly sampled, but only that they belong to some pre-defined set R^{LPKE}.

6.2 An R_{BM}-Lossy PKE Scheme from LWE

In this section we describe an R_{BM}-lossy PKE scheme below. Our scheme builds on a variant of the primal Regev cryptosystem [Reg05] suggested in [GPV08].

Definition 6.11. *Let $\mathcal{K} = \{0,1,\perp\}^L$ and $\mathcal{T} = \{0,1\}^\ell$, for some $\ell, L \in \mathsf{poly}(\lambda)$ such that $\ell < L$. Let F_{ADH} the partitioning function defined by $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$ in Definition 2.1. The **bit-matching relation** $R_{\mathsf{BM}} : \mathcal{K} \times \mathcal{T} \to \{0,1\}$ for AHF is the relation where $R_{\mathsf{BM}}(K,t) = 1$ if and only if $K = K_1 \dots K_L$ and $t = t_1 \dots t_\ell$ satisfy $F_{\mathsf{ADH}}(K,t) = 0$ (namely, $\bigwedge_{i=1}^{L}(K_i = \perp) \vee (K_i = \mathsf{AHF}(t)_i)$).*

6.2.1 The Construction

Let $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$ an admissible hash function (Definition 2.1) with key space $\mathcal{K} = \{0,1,\perp\}^L$ and let $R_{\mathsf{BM}} \subset \mathcal{K} \times \{0,1\}^\ell$ the corresponding bit-matching relation. We construct an R_{BM}-lossy PKE scheme in the following way.

Par-Gen(1^λ)**:** Given a security parameter $\lambda \in \mathbb{N}$, let $n_0 = \mathsf{poly}(\lambda)$ the length of messages. Choose a prime modulus $q = \mathsf{poly}(\lambda)$; dimensions $n = n_0 + \Omega(\lambda)$ and $m = 2n\lceil \log q \rceil + O(\lambda)$. Define the tag space as $\mathcal{T} = \{0,1\}^\ell$ where $\ell = \Theta(\lambda)$. Define the initialization value space $\mathcal{K} = \{0,1,\perp\}^L$ and Gaussian parameters $\sigma = O(m) \cdot L$ and $\alpha \in (0,1)$ such that $m^2 \alpha q \cdot (L+1) \cdot \sigma\sqrt{2m} < q/4$. Define public parameters as $\Gamma = (\ell, L, n_0, q, n, m, u, \alpha, \sigma)$.

Keygen(Γ, K)**:** On input of public parameters Γ and an initialization value $K \in \{0,1,\perp\}^L$, generate a key pair as follows.

1. Sample random matrices $\bar{\mathbf{B}} \leftarrow\!\!\$\ U(\mathbb{Z}_q^{(n-n_0)\times m})$, $\mathbf{S} \leftarrow\!\!\$\ U(\mathbb{Z}_q^{(n-n_0)\times n_0})$ and a small-norm $\mathbf{E} \leftarrow\!\!\$\ D_{\mathbb{Z}^{m\times n_0}, \alpha q}$ to compute
$$\mathbf{A} = \begin{bmatrix} \bar{\mathbf{B}} \\ \mathbf{S}^\top \cdot \bar{\mathbf{B}} + \mathbf{E}^\top \end{bmatrix} \in \mathbb{Z}_q^{n\times m}.$$
2. Parse K as $K_1 \dots K_L \in \{0,1,\perp\}^L$. Letting $\mathbf{G} \in \mathbb{Z}_q^{n\times m}$ denote the gadget matrix, for each $i \in [L]$ and $b \in \{0,1\}$, compute matrices $\mathbf{A}_{i,b} \in \mathbb{Z}_q^{n\times m}$ as
$$\mathbf{A}_{i,b} = \begin{cases} \mathbf{A}\cdot\mathbf{R}_{i,b} + \mathbf{G} & \text{if } (K_i \neq \perp) \wedge (b = 1 - K_i) \\ \mathbf{A}\cdot\mathbf{R}_{i,b} & \text{if } (K_i = \perp) \vee (b = K_i). \end{cases} \tag{6.1}$$
where $\mathbf{R}_{i,b} \leftarrow\!\!\$\ U(\{-1,1\}^{m\times m})$ for all $i \in [L]$ and $b \in \{0,1\}$.

Define $R^{\mathsf{LPKE}} = \{\mathbf{r} \in \mathbb{Z}^{2m} \mid \|\mathbf{r}\| \leq \sigma\sqrt{2m}\}$ and output $sk = (K, \mathbf{S})$ as well as
$$pk := \left(\mathbf{A},\ \{\mathbf{A}_{i,b}\}_{(i,b)\in[L]\times\{0,1\}}\right), \qquad tk = (K, \{\mathbf{R}_{i,b}\}_{(i,b)\in[L]\times\{0,1\}}).$$

LKeygen(Γ, K)**:** This algorithm proceeds identically to Keygen except that steps 1 and 2 are modified in the following way.

1. Run $(\mathbf{A}, \mathbf{T}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^m, 1^n)$ so as to obtain a statistically uniform matrix $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$ with a trapdoor for the lattice $\Lambda^{\perp}(\mathbf{A})$ (Lemma 2.8). Notice $m = 2n\lceil \log q \rceil + O(\lambda)$ is required by Lemma 2.8 in order to run algorithm TrapGen.
2. Define matrices $\{\mathbf{A}_{i,b} \in \mathbb{Z}_q^{n\times m}\}_{(i,b)\in[L]\times\{0,1\}}$ as in (6.1).

Define R^{LPKE} as in Keygen and output

$$pk := \Big(\mathbf{A},\ \{\mathbf{A}_{i,b}\}_{(i,b)\in[L]\times\{0,1\}}\Big), \quad sk = \mathbf{T}_{\mathbf{A}}, \quad tk = (K, \{\mathbf{R}_{i,b}\}_{(i,b)\in[L]\times\{0,1\}}).$$

Encrypt(pk, t, Msg)**:** To encrypt $\mathsf{Msg} \in \{0,1\}^{n_0}$ for the tag $t = t_1 \dots t_\ell \in \{0,1\}^\ell$, conduct the following steps.

1. Encode the tag t as $t' = t'_1 \dots t'_L = \mathsf{AHF}(t) \in \{0,1\}^L$ and compute $\mathbf{A}_{F,t} = \sum_{i=1}^{L} \mathbf{A}_{i,t'_i} \in \mathbb{Z}_q^{n\times m}$. Note that $\mathbf{A}_{F,t} = \mathbf{A}\cdot\mathbf{R}_{F,t} + d_t\cdot\mathbf{G}$ for some $\mathbf{R}_{F,t} \in \mathbb{Z}^{m\times m}$ of norm $\|\mathbf{R}_{F,t}\|_\infty \le L\cdot m$ and where $d_t \in \{0,\dots,L\}$ is the number of non-$\perp$ entries of K for which $K_i \neq t'_i$.
2. Choose $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and output $\perp$ if $\mathbf{r} \notin R^{\mathsf{LPKE}}$. Otherwise, output

$$\mathbf{c} = [\mathbf{A} \,|\, \mathbf{A}_{F,t}]\cdot\mathbf{r} + \begin{bmatrix} \mathbf{0}^{n-n_0} \\ \mathsf{Msg}\cdot\lfloor q/2\rfloor \end{bmatrix} \in \mathbb{Z}_q^n \ . \tag{6.2}$$

Decrypt$(sk, t, \mathbf{c})$**:** Given $sk = (K, \mathbf{S})$ and the tag $t \in \{0,1\}^\ell$, compute $t' = t'_1 \dots t'_L = \mathsf{AHF}(t) \in \{0,1\}^L$ and return $\perp$ if $R_{\mathsf{BM}}(K, t') = 0$. Otherwise, compute $\mathbf{w} = [-\mathbf{S}^\top \,|\, \mathbf{I}_{n_0}]\cdot\mathbf{c} \in \mathbb{Z}_q^{n_0}$. For each $i \in [n_0]$, do the following:

1. If neither $\mathbf{w}[i]$ nor $|\mathbf{w}[i] - \lfloor q/2\rfloor|$ is close to 0, halt and return $\perp$.
2. Otherwise, set $\mathsf{Msg}[i] \in \{0,1\}$ so as to minimize $|\mathbf{w}[i] - \mathsf{Msg}[i]\cdot\lfloor q/2\rfloor|$.

Return $\mathsf{Msg} = \mathsf{Msg}[1]\dots\mathsf{Msg}[n_0]$.

Opener$(pk, tk, t, \mathbf{c}, \mathsf{Msg}_1)$**:** Given $tk = (K, \{\mathbf{R}_{i,b}\}_{i,b})$ and $t \in \{0,1\}^\ell$, compute $t' = t'_1 \dots t'_L = \mathsf{AHF}(t) \in \{0,1\}^L$ and return $\perp$ if $R_{\mathsf{BM}}(K, t') = 1$. Otherwise,

1. Compute the small-norm matrix $\mathbf{R}_{F,t} = \sum_{i=1}^{L} \mathbf{R}_{i,t'_i} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{A}_{F,t} = \mathbf{A}\cdot\mathbf{R}_{F,t} + d_t\cdot\mathbf{G}$ and $\|\mathbf{R}_{F,t}\|_\infty \le L\cdot m$ with $d_t \in [L]$.

2. Use $\mathbf{R}_{F,t} \in \mathbb{Z}^{m\times m}$ as a trapdoor for the matrix

$$\bar{\mathbf{A}}_{F,t} = [\mathbf{A} \mid \mathbf{A}_{F,t}] = [\mathbf{A} \mid \mathbf{A}\cdot\mathbf{R}_{F,t} + d_t \cdot \mathbf{G}] \in \mathbb{Z}_q^{n\times 2m}$$

to sample a Gaussian vector $\bar{\mathbf{r}} \in \mathbb{Z}^{2m}$ such that

$$\bar{\mathbf{A}}_{F,t}\cdot\bar{\mathbf{r}} = \mathbf{c} - \begin{bmatrix} \mathbf{0}^{n-n_0} \\ \mathsf{Msg}_1 \cdot \lfloor q/2 \rfloor \end{bmatrix} . \tag{6.3}$$

Namely, defining $\mathbf{c}_{\mathsf{Msg}_1} = \mathbf{c} - [(\mathbf{0}^{n-n_0})^\top \mid \mathsf{Msg}_1^\top \cdot \lfloor q/2 \rfloor]^\top$, sample and output fake random coins $\bar{\mathbf{r}} \hookleftarrow D_{\Lambda_q^{\mathbf{c}_{\mathsf{Msg}_1}}(\bar{\mathbf{A}}_{F,t}),\sigma}$.

LOpener$(sk, t, \mathbf{c}, \mathsf{Msg}_1)$**:** Given $sk = \mathbf{T}_\mathbf{A}$ and $t \in \{0,1\}^\ell$, use $\mathbf{T}_\mathbf{A}$ to derive a trapdoor $\mathbf{T}_{\mathbf{A},t}$ for the lattice $\Lambda_q^\perp(\bar{\mathbf{A}}_{F,t})$ and use $\mathbf{T}_{\mathbf{A},t}$ to sample a Gaussian vector $\bar{\mathbf{r}} \hookleftarrow D_{\Lambda_q^{\mathbf{c}_{\mathsf{Msg}_1}}(\bar{\mathbf{A}}_{F,t}),\sigma}$ satisfying (6.3).

6.2.2 Security

The Theorem below states that the construction has the required properties under the LWE assumption.

Theorem 6.1. *The above construction is an R_{BM}-lossy public-key encryption scheme with efficient opening under the* LWE *assumption.*

Proof. To prove the statement, we prove that the scheme enables correct decryption with overwhelming probability in injective mode. We also prove the indistinguishability properties using the LWE assumption on one occasion.

Decryption under injective tags. For any initialization value $K \in \mathcal{K}$, any injective tag $t \in \{0,1\}^\ell$ (i.e. $(K,t) \in R_{\mathsf{BM}}$), any message $\mathsf{Msg} \in \{0,1\}^{n_0}$, and any encryption $\mathbf{c} \in \mathbb{Z}_q^n$ of Msg under the $\mathsf{pk} = \left(\mathbf{A}, \{\mathbf{A}_{i,b}\}_{i,b}\right)$ and t, we have:

$$[-\mathbf{S}^\top \mid \mathbf{I}_{n_0}]\cdot\mathbf{c} = \mathbf{E}^\top\cdot[\mathbf{I}_m \mid \mathbf{R}_{F,t}]\cdot\mathbf{r} + \mathsf{Msg}\cdot\lfloor q/2\rfloor \in \mathbb{Z}_q^{n_0}$$

We show that, for any $\mathbf{r} \in \mathbb{Z}^{2m}$ of norm smaller than $\|\mathbf{r}\|_\infty \le \|\mathbf{r}\| \le \sigma\sqrt{2m}$, we have $\|\mathbf{E}^\top[\mathbf{I}_m \mid \mathbf{R}_{F,t}]\cdot\mathbf{r}\|_\infty < q/4$ with overwhelming probability over the randomness of Keygen, so that the decryption algorithm recovers the message. To prove this, notice that our definition of the randomness space R^{LPKE} imposes $\|\mathbf{r}\|_\infty \le \|\mathbf{r}\| \le \sigma\sqrt{2m}$. Besides, we also have

$$\|[\mathbf{I}_m \mid \mathbf{R}_{F,t}]\|_\infty \le 1 + \|\mathbf{R}_{F,t}\|_\infty \le 1 + L\cdot m$$

with probability 1 and

$$\|\mathbf{E}^\top\|_\infty = \max_{i\in[n_0]} \sum_{j=1}^{m} |e_{ij}| \le \sqrt{m}\cdot\max_{i\in[n_0]} \sqrt{\sum_{j=1}^{m} e_{ij}^2} \le m\cdot\alpha q$$

with overwhelming probability when $\mathbf{E}^\top \hookleftarrow D_{\mathbb{Z}^{n_0\times m},\alpha q}$. Putting the above altogether, our choice of parameters implies that

$$\|\mathbf{E}^\top\|_\infty \cdot \|[\mathbf{I}_m \mid \mathbf{R}_{F,t}]\|_\infty \cdot \|\mathbf{r}\|_\infty \leq m\alpha q \cdot (L\cdot m+1)\cdot \sigma\sqrt{2m} < q/4 \ .$$

Indistinguishability. The key generation algorithm $\mathsf{LKeygen}$ and Keygen satisfy the following properties:

(i) The LWE assumption implies that, for any $K \in \mathcal{K}_\lambda$, the distributions $D_{\text{loss}} = \{(pk,tk) \mid (pk,sk,tk) \leftarrow \mathsf{LKeygen}(\Gamma,K)\}$ and $D_{\text{inj}} = \{(pk,tk) \mid (pk,sk,tk) \leftarrow \mathsf{Keygen}(\Gamma,K)\}$ are computationally indistinguishable. These distributions only differ in the generation of the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. The matrix $\mathbf{A}$ produced by the Keygen algorithm is pseudo-random since, under the $\mathsf{LWE}_{q,m,n-n_0,\alpha}$ assumption, we can replace $\mathbf{S}^\top\bar{\mathbf{B}} + \mathbf{E}^\top$ by a uniform matrix $\mathbf{B} \sim U(\mathbb{Z}_q^{n_0\times m})$ without the adversary noticing. When using $\mathsf{LKeygen}$, the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is statistically uniform by the properties of the $\mathsf{TrapGen}$ algorithm (specifically, Lemma 2.8).

(ii) For any distinct initialization values $K, K' \in \mathcal{K}_\lambda$, the two distributions $\{pk \mid (pk,sk,tk) \leftarrow \mathsf{LKeygen}(\Gamma,K)\}$ and $\{pk \mid (pk,sk,tk) \leftarrow \mathsf{LKeygen}(\Gamma,K')\}$ are statistically indistinguishable since the public matrices $(\mathbf{A},\{\mathbf{A}_{i,b}\})$ are statistically uniform and independent regardless of which K is used as input by $\mathsf{LKeygen}$. Recall the matrix $\mathbf{A}$ produced by the $\mathsf{LKeygen}$ algorithm is statistically close to $U(\mathbb{Z}_q^{n\times m})$ by the properties of the $\mathsf{TrapGen}$. As for the matrices $\mathbf{A}_{i,b} = \mathbf{A}\cdot\mathbf{R}_{i,b} + \kappa_{i,b}\cdot\mathbf{G}$, for $\kappa_{i,b} \in \{0,1\}$, the Leftover Hash Lemma 2.18 implies that the statistical distance between the distributions $\{(\mathbf{A},\mathbf{A}\cdot\mathbf{R}_{i,b}) \mid \mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{n\times m}), \mathbf{R}_{i,b} \hookleftarrow U(\{-1,1\}^{m\times m})\}$ and $\{(\mathbf{A},\mathbf{A}_{i,b}) \mid \mathbf{A},\mathbf{A}_{i,b} \hookleftarrow U(\mathbb{Z}_q^{n\times m})\}$ is smaller than $m\cdot\sqrt{q^n/2^m} < 2^{-\lambda}$, where the last inequality is implied by our choice of $m = 2n\lceil\log q\rceil + O(\lambda)$.

Lossiness under lossy tags. It is enough to prove that the distribution of a ciphertext obtained by encrypting under a lossy tag is statistically close to the uniform distribution on $\mathbb{Z}_q^n$.

For any initialization value $K \in \mathcal{K}_\lambda$ and tag $t \in \{0,1\}^\ell$ such that $(K,t) \notin R_{\mathsf{BM}}$, any $\big(pk = (\mathbf{A},\{\mathbf{A}_{i,b}\}_{i\in[L],b\in\{0,1\}}), sk = (\mathbf{S},K), tk\big) \leftarrow \mathsf{Keygen}(\Gamma,K)$, and any message $\mathsf{Msg} \in \{0,1\}^{n_0}$, an encryption of Msg is generated as

$$\mathbf{c} = [\mathbf{A} \mid \mathbf{A}_{F,t}]\cdot\mathbf{r} + \begin{bmatrix}\mathbf{0}^{n-n_0}\\ \mathsf{Msg}\cdot\lfloor q/2\rfloor\end{bmatrix} \in \mathbb{Z}_q^n \ . \tag{6.4}$$

where $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and $\bar{\mathbf{A}}_{F,t} = [\mathbf{A} \mid \mathbf{A}\cdot\mathbf{R}_{F,t} + d_t\cdot\mathbf{G}] \in \mathbb{Z}_q^{n\times 2m}$. The matrix $\bar{\mathbf{A}}_{F,t}$ is of this form, with $d_t > 0$, because t is a lossy tag (i.e., $(K,t) \notin R_{\mathsf{BM}}$). This implies that the columns of $\bar{\mathbf{A}}_{F,t}$ generate $\mathbb{Z}_q^n$. By [MP12, Lemma 5.3], we know that $\bar{\mathbf{A}}_{F,t}$ has a trapdoor $\mathbf{T}_{F,t} \in \mathbb{Z}^{2m\times 2m}$ (namely, a short basis of the lattice $\Lambda^\perp(\bar{\mathbf{A}}_{F,t})$) such that

$\|\tilde{\mathbf{T}}_{F,t}\| \leq (\|\mathbf{R}_{F,t}\|+1)\cdot\sqrt{5}$ and thus $\|\tilde{\mathbf{T}}_{F,t}\| \leq \sqrt{5}\cdot(\sqrt{m}\cdot L+1)$. Again, by [GPV08, Lemma 3.1], we know that $\eta_{2^{-m}}(\Lambda^{\perp}(\bar{\mathbf{A}}_{F,t})) \leq \|\tilde{\mathbf{T}}_{F,t}\| \cdot O(\sqrt{m})$. By choosing $\sigma = O(m)\cdot L$, we have $\sigma \geq \eta_{2^{-m}}(\Lambda^{\perp}(\bar{\mathbf{A}}_{F,t}))$. By applying [GPV08, Lemma 5.2], we conclude that $\bar{\mathbf{A}}_{F,t}\cdot\mathbf{r}$ is statistically close to the uniform distribution $U(\mathbb{Z}_q^n)$ when $\mathbf{r} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$.

Efficient opening under lossy tags. From the previous paragraph, we know that the lattice $\Lambda_q^{\perp}(\bar{\mathbf{A}}_{F,t})$ has a basis satisfying $\|\tilde{\mathbf{T}}_{F,t}\| \leq \sqrt{5}\cdot(\sqrt{m}L+1)$. By the choice of $\sigma = O(m)\cdot L$, the condition $\sigma \geq \|\tilde{\mathbf{T}}_{F,t}\|\cdot\omega(\sqrt{\log 2m})$ holds. For any $\mathbf{c}_{\mathsf{Msg}_1} \in \mathbb{Z}_q^n$, we can thus apply Lemma 2.2 and sample a Gaussian vector $\bar{\mathbf{r}} \in \mathbb{Z}^{2m}$ from the distribution $D_{\Lambda_q^{\mathbf{c}_{\mathsf{Msg}_1}}(\bar{\mathbf{A}}_{F,t}),\sigma}$. Our argument to prove the lossiness under lossy tags implies that encrypting any message $\mathsf{Msg}_0 \in \{0,1\}^{n_0}$ under a lossy tag leads to a statistically uniform ciphertext $\mathbf{c} \sim_s U(\mathbb{Z}_q^n)$. In particular, for any $\mathsf{Msg}_1 \in \{0,1\}^{n_0}$, the distribution

$$\{\big(\bar{\mathbf{A}}_{F,t},\ \mathbf{c}_{\mathsf{Msg}_1} = \bar{\mathbf{A}}_{F,t}\cdot\mathbf{r}_0 + [\tfrac{\mathbf{0}^{n-n_0}}{\mathsf{Msg}_0\cdot\lfloor q/2\rfloor}] - [\tfrac{\mathbf{0}^{n-n_0}}{\mathsf{Msg}_1\cdot\lfloor q/2\rfloor}],\ \bar{\mathbf{r}}\big)\ \mid \mathbf{r}_0 \hookleftarrow D_{\mathbb{Z}^{2m},\sigma},\ \bar{\mathbf{r}} \hookleftarrow D_{\Lambda^{\mathbf{c}_{\mathsf{Msg}_1}}(\bar{\mathbf{A}}_{F,t}),\sigma}\}$$

is statistically close to

$$\left\{\big(\bar{\mathbf{A}}_{F,t},\ \mathbf{c}_{\mathsf{Msg}_1} = \mathbf{c} - [\tfrac{\mathbf{0}^{n-n_0}}{\mathsf{Msg}_1\cdot\lfloor q/2\rfloor}],\ \bar{\mathbf{r}}\big) \mid \mathbf{c} \hookleftarrow U(\mathbb{Z}_q^n),\ \bar{\mathbf{r}} \hookleftarrow D_{\Lambda^{\mathbf{c}_{\mathsf{Msg}_1}}(\bar{\mathbf{A}}_{F,t}),\sigma}\right\}\ ,$$

which is itself statistically close to $\left\{\big(\bar{\mathbf{A}}_{F,t},\ \mathbf{c}_{\mathsf{Msg}_1} = \bar{\mathbf{A}}_{F,t}\cdot\mathbf{r},\ \mathbf{r}\big) \mid\ \mathbf{r} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}\right\}$.

Efficient opening under lossy keys. By [CHKP10, Lemma 3.2], we know that a basis $\mathbf{T}_{\mathbf{A},t} \in \mathbb{Z}^{2m\times 2m}$ for the lattice $\Lambda_q^{\perp}([\mathbf{A}|\mathbf{A}_{F,t}])$ can be efficiently computed given a basis $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m\times m}$ of the lattice $\Lambda_q^{\perp}(\mathbf{A})$. Moreover, this basis satisfies $\|\tilde{\mathbf{T}}_{\mathbf{A}}\| = \|\tilde{\mathbf{T}}_{\mathbf{A},t}\|$. By Lemma 2.8, it follows that $\|\tilde{\mathbf{T}}_{\mathbf{A},t}\| \leq O(\sqrt{n\log q}) = O(\sqrt{m})$. By the choice of parameters, we obtain that $\sigma \geq \|\tilde{\mathbf{T}}_{\mathbf{A},t}\|\cdot\omega(\sqrt{\log 2m})$. Hence, by Lemma 2.2, we can sample $\bar{\mathbf{r}} \in \mathbb{Z}^{2m}$ from a distribution statistically close to $D_{\Lambda_q^{\mathbf{c}_{\mathsf{Msg}_1}}(\bar{\mathbf{A}}_{F,t}),\sigma}$. The claim follows from the same arguments as in the case of efficient openings under lossy tags.

□

6.3 Direct Constructions of Multi-Theorem NIZK from Trapdoor Sigma-protocols

In this section we present a generic transformation from trapdoor Σ-protocols to multi-theorem NIZKs. We need to assume lossy encryption with efficient opening and correlation-intractable hash functions.

6.3.1 Overview

Our multi-theorem compiler is inspired by the modified protocol of [CLW19] for proving graph Hamiltonicity [FLS99]. We quickly recall the ideas of [CLW19] below. The prover's first message $\mathbf{a}$ of the interactive protocol is computed using a

lossy encryption scheme, instead of an usual commitment. To make the scheme non-interactive, the random oracle of the Fiat-Shamir transform is replaced by a correlation-intractable hash function. The hash is applied to the transcript of the protocol so far, to derive the challenge $\mathsf{Chall} = h(k,(x,\mathbf{a}))$. The proof for the message x is given by $\boldsymbol{\pi} := (\mathbf{a},\mathsf{Chall},\mathbf{z})$, where $\mathbf{z}$ is the final response of the prover of the interactive Σ-protocol.

Recall that, depending on the generation of the keys, a lossy encryption behaves either as an extractable commitment or as a statistically-hiding commitment. The extractable mode is used to prove soundness, while the hiding property is used to prove zero-knowledge.

To simulate a proof, the honest verifier simulator is invoked on a random challenge, to produce a transcript $(\mathbf{a},\mathsf{Chall},\mathbf{z}) \leftarrow \mathsf{HVZK}(x,\mathsf{Chall})$. Because the correlation intractable hash family can be programmed, they are able to generate a key k such that the challenge is explained as the hash value $\mathsf{Chall} = h(k,(x,\mathbf{a}))$. This simulation strategy cannot be employed to prove multiple theorems, because for each new proof they would need a new programmed key for the hash function. This implies a CRS size proportional to the number of proofs that the adversary queries.

To simulate multiple proofs while keeping the CRS size constant, we notice that some particular lossy encryptions can admit efficient opening algorithms, that allow explaining a ciphertext as the encryption of any message of our choice.

In our construction the prover's first message $\mathbf{a}' \leftarrow P(\mathsf{crs},x,w)$ is encrypted by a lossy encryption scheme as $\mathbf{a} \leftarrow \mathsf{Encrypt}(pk,\mathbf{a}')$ and the challenge is computed as the hash $\mathsf{Chall} := h(k,(x,\mathbf{a}))$ of the message and the encryption of the prover's message.

We simulate a proof by first computing the challenge $\mathsf{Chall} := h(k,(x,\mathbf{a}))$, for a dummy lossy encryption $\mathbf{a} = \mathsf{Encrypt}(pk,\mathbf{0})$. Now we can invoke the honest verifier simulator to obtain a transcript $(\mathbf{a}',\mathsf{Chall},\mathbf{z}') \leftarrow \mathsf{HVZK}(x,\mathsf{Chall})$. The efficient opening algorithm returns some encryption randomness $\mathbf{r}$ such that $\mathbf{a} = \mathsf{Encrypt}(pk,\mathbf{a}';\mathbf{r})$, i.e. the ciphertext $\mathbf{a}$ is explained as an encryption of the prover's message $\mathbf{a}'$. This allows us to avoid programming the hashing keys, and we are able to prove zero-knowledge when unbounded number of proofs are queried by the adversary.

6.3.2 Direct Construction

In order to compile trapdoor Σ-protocols into multi-theorem NIZK proof systems for the same language, we use the following building blocks.

- A trapdoor Σ-protocol $\Pi' = (\mathsf{Gen}'_{\mathsf{par}},\mathsf{Gen}'_{\mathscr{L}},\mathsf{P}',\mathsf{V}')$ with challenge space ChSp, for a language $\mathscr{L} = (\mathscr{L}_{\mathsf{zk}},\mathscr{L}_{\mathsf{sound}})$. In addition, $\mathsf{BadChallenge}(\tau_\Sigma,\mathsf{crs},x,\mathbf{a})$ should be computable by a boolean circuit of size $S \in \mathsf{poly}(\lambda)$ for any input $(\tau_\Sigma,\mathsf{crs},x,\mathbf{a})$.

- A somewhere correlation intractable hash family $\mathscr{H} = (\mathsf{Gen}, \mathsf{Hash})$ with output in ChSp for the class $\mathscr{R}_{\mathsf{CI}}$ of relations that are efficiently searchable in size $2S \in \mathsf{poly}(\lambda)$.
- A lossy PKE scheme $\Pi^{\mathsf{LPKE}} = (\mathsf{Par\text{-}Gen}, \mathsf{Keygen}, \mathsf{LKeygen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Opener})$ with public (resp. secret) key space $\mathscr{PK}$ (resp. $\mathscr{SK}$), as defined in Section 6.1.5. We assume that the decryption algorithm Decrypt is computable by a boolean circuit of size S. We denote the message (resp. ciphertext) space by MsgSp (resp. CtSp) and the randomness space by R^{LPKE}. Let also D_R^{LPKE} denote the distribution from which the random coins of Encrypt are sampled.

Our construction $\Pi = (\mathsf{Gen}_{\mathsf{par}}, \mathsf{Gen}_{\mathscr{L}}, \mathsf{P}, \mathsf{V})$ goes as follows.

$\mathsf{Gen}_{\mathsf{par}}(1^\lambda)$: Run $\mathsf{par} \leftarrow \mathsf{Gen}'_{\mathsf{par}}(1^\lambda)$ and output par.

$\mathsf{Gen}_{\mathscr{L}}(\mathsf{par}, \mathscr{L})$: Given public parameters par and a language $\mathscr{L} = (\mathscr{L}_{\mathsf{zk}}, \mathscr{L}_{\mathsf{sound}})$, generate the common reference string as follows.

1. Generate a common reference string $\mathsf{crs}'_{\mathscr{L}} \leftarrow \mathsf{Gen}'_{\mathscr{L}}(\mathsf{par}, \mathscr{L})$ for the trapdoor Σ-protocol Π'.
2. Generate a key $k \leftarrow \mathsf{Gen}(1^\lambda)$ for the somewhere correlation intractable hash function.
3. Generate public parameters $\Gamma \hookleftarrow \mathsf{Par\text{-}Gen}(1^\lambda)$ for the lossy PKE scheme Π^{LPKE}. Then, generate lossy keys $(pk, sk) \leftarrow \mathsf{LKeygen}(\Gamma)$.

Output the language-dependent $\mathsf{crs}_{\mathscr{L}} := \big(\mathsf{crs}'_{\mathscr{L}}, k\big)$. The global common reference string consists of

$$\mathsf{crs} = \big(\mathsf{par}, \mathsf{crs}_{\mathscr{L}}, \Gamma, pk\big).$$

$\mathsf{P}(\mathsf{crs}, x, w)$: To prove a statement x using a witness $w \in R_{\mathsf{zk}}(x)$,

1. Compute $\big(\mathbf{a}', st\big) \leftarrow \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}}, x, w)$ via the invocation of the prover algorithm of Π'. Compute $\mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathbf{a}'; \mathbf{r})$ using randomness $\mathbf{r} \hookleftarrow D_R^{\mathsf{LPKE}}$ sampled from the distribution D_R^{LPKE} over R^{LPKE}.
2. Compute $\mathsf{Chall} := \mathsf{Hash}(k, (x, \mathbf{a}))$
3. Compute $\mathbf{z}' = \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}}, x, w, \mathbf{a}, \mathsf{Chall}, st)$
4. Output the proof $\boldsymbol{\pi} = \big(\mathbf{a}', \mathbf{z}', \mathbf{r}\big)$.

$\mathsf{V}(\mathsf{crs}, x, \boldsymbol{\pi})$: Given a statement x and a candidate proof $\boldsymbol{\pi} = \big(\mathbf{a}', \mathbf{z}', \mathbf{r}\big)$, do the following. Compute $\mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathbf{a}'; \mathbf{r})$. Compute the challenge $\mathsf{Chall} = \mathsf{Hash}\big(k, (x, \mathbf{a})\big)$. If $\mathsf{V}'(\mathsf{crs}'_{\mathscr{L}}, x, (\mathbf{a}', \mathsf{Chall}, \mathbf{z}')) = 1$ if, return 1. Otherwise, return 0.

6.3.3 Security Proofs for the Multi-Theorem NIZK

We show below that the properties of Definition 6.3 are satisfied by the previous construction.

Theorem 6.2. *The above argument system is multi-theorem (statistically) ZK assuming that the trapdoor Σ-protocol Π' is special (statistically) ZK.*

Proof. We describe a zero-knowledge simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ that uses the lossy secret key $\tau_{\mathrm{zk}} = sk$ of Π^{LPKE} to generate proofs $\boldsymbol{\pi} = (\mathbf{a}', \mathbf{z}', \mathbf{r})$ without using the witnesses. Namely, on input of $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$, Sim_0 generates $\mathsf{crs}_{\mathscr{L}}$ by proceeding identically to $\mathsf{Gen}_{\mathscr{L}}$ while Sim_1 is described hereunder.

$\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, x)$: Given a statement x and the simulation trapdoor $\tau_{\mathrm{zk}} = sk$, algorithm Sim_1 proceeds as follows.

1. Let $\mathbf{0}^{|\mathbf{a}'|}$ the all-zeroes string of the same length as the first prover message of Π'. Compute
$$\mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathbf{0}^{|\mathbf{a}'|}; \mathbf{r}_0)$$
using random coins $\mathbf{r}_0 \hookleftarrow D_R^{\mathsf{LPKE}}$ sampled from the distribution D_R^{LPKE}.
2. Compute $\mathsf{Chall} = \mathsf{Hash}(k, (x, \mathbf{a}))$.
3. Run the ZK simulator $(\mathbf{a}', \mathbf{z}') \leftarrow \mathsf{ZKSim}(\mathsf{crs}'_{\mathscr{L}}, x, \mathsf{Chall})$ of Π' so as to obtain a simulated transcript $(\mathbf{a}', \mathsf{Chall}, \mathbf{z}')$ of Π' for the challenge Chall.
4. Using the lossy secret key sk of Π^{LPKE}, compute random coins $\mathbf{r} \leftarrow \mathsf{Opener}(pk, sk, \mathbf{a}, \mathbf{a}')$ that explain $\mathbf{a}$ as an encryption of $\mathbf{a}'$. Then, output the proof $\boldsymbol{\pi} = (\mathbf{a}', \mathbf{z}', \mathbf{r})$.

We now prove that the simulation is statistically indistinguishable from proofs generated by the real prover. The special property of Π' implies that its simulator produces $(\mathbf{a}', \mathbf{z}') \leftarrow \mathsf{ZKSim}(\mathsf{crs}'_{\mathscr{L}}, x, \mathsf{Chall})$ such that $(\mathbf{a}', \mathsf{Chall}, \mathbf{z}')$ is indistinguishable from a real transcript with challenge Chall. This implies that the distribution

$$\{(\mathbf{a}, \mathbf{a}', \mathbf{r}, \mathbf{z}') \mid \mathbf{r}_0 \hookleftarrow D_R^{\mathsf{LPKE}},\ \mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathbf{0}^{|\mathbf{a}'|}; \mathbf{r}_0),\ (\mathbf{a}', \mathbf{z}') \leftarrow \mathsf{ZKSim}(\mathsf{crs}'_{\mathscr{L}}, x, \mathsf{Chall}),\ \mathbf{r} \leftarrow \mathsf{Opener}(pk, sk, \mathbf{a}, \mathbf{a}')\}, \tag{6.5}$$

is computationally indistinguishable from

$$\{(\mathbf{a}, \mathbf{a}', \mathbf{r}, \mathbf{z}') \mid \mathbf{r}_0 \hookleftarrow D_R^{\mathsf{LPKE}},\ \mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathbf{0}^{|\mathbf{a}'|}; \mathbf{r}_0),\ (\mathbf{a}', st') \leftarrow \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}}, x, w),\ \mathbf{z}' = \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}}, x, w, \mathbf{a}', \mathsf{Chall}, st'),\ \mathbf{r} \leftarrow \mathsf{Opener}(pk, sk, \mathbf{a}, \mathbf{a}')\}.$$

By the property of Lossiness under lossy keys of the Π^{LPKE} ciphertexts, the above is statistically close to:

$$\{(\mathbf{a},\mathbf{a}',\mathbf{r},\mathbf{z}') \quad | \quad \mathbf{r}_0 \hookleftarrow D_R^{\mathsf{LPKE}},\ \mathbf{a} \leftarrow \mathsf{Encrypt}(pk,\mathbf{a}';\mathbf{r}_0),\ (\mathbf{a}',st') \leftarrow \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}},x,w),\ \mathbf{z}' = \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}},x,w,\mathbf{a}',\mathsf{Chall},st'),\ \mathbf{r} \leftarrow \mathsf{Opener}(pk,sk,\mathbf{a},\mathbf{a}')\}.$$

By the property of efficient opening under lossy keys, we know that the above is statistically indistinguishable from

$$\{(\mathbf{a},\mathbf{a}',\mathbf{r},\mathbf{z}') \quad | \quad (\mathbf{a}',st') \leftarrow \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}},x,w),\ \mathbf{r} \hookleftarrow D_R^{\mathsf{LPKE}},\ \mathbf{a} \leftarrow \mathsf{Encrypt}(pk,\mathbf{a}';\mathbf{r}),\ \mathbf{z}' = \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}},x,w,\mathbf{a}',\mathsf{Chall},st')\}. \tag{6.6}$$

The distribution (6.5) corresponds to proofs generated by the simulator Sim_1, while (6.6) is identical to the distribution generated by the real prover. Simulated proofs are thus indistinguishable from real proofs if the simulator of Π' is ZK. Notice that if the simulator is statistically ZK then this implies that the simulated proofs are actually statistically indistinguishable from real proofs.

□

Theorem 6.3. *The above argument system provides non-adaptive soundness assuming that: (i)* Π^{LPKE} *is a lossy encryption scheme; (ii) The hash family* $\mathscr{H}$ *is somewhere correlation-intractable for all relations that are searchable in size* $2S$*, where* S *denotes the size of a boolean circuit computing* $\mathsf{BadChallenge}(\cdot,\cdot,\cdot,\cdot)$ *and the size of a boolean decryption circuit.*

Proof. To prove the result, we consider a sequence of games. For each i, we define a Boolean variable $W_i \in \{\mathtt{true},\mathtt{false}\}$ where $W_i = \mathtt{true}$ if and only if the adversary wins in Game_i.

Game_0: This is the real soundness experiment. Namely, the challenger generates the parameters as in the real scheme and gives $\mathsf{crs} = (\mathsf{par},\mathsf{crs}_{\mathscr{L}},\Gamma,pk)$ to the adversary $\mathscr{A}$ and a statement $x^\star \notin \mathscr{L}_{\mathsf{sound}}$. The adversary outputs a proof $\pi^\star$, and W_0 is set to 1 iff $\mathsf{V}(\mathsf{crs},x^\star,\pi^\star) = 1$. By the definition of $\mathscr{A}$'s advantage, we have $\mathbf{Adv}^{\text{soundness}}_{\mathscr{A}}(\lambda) = \Pr[W_0]$.

Game_1: We change the distribution of $\mathsf{crs}_{\mathscr{L}} = (\mathsf{crs}'_{\mathscr{L}},k)$ by leveraging the CRS indistinguishability property of the trapdoor Σ-protocol Π'. Namely, we use the $\mathsf{TrapGen}'$ algorithm of Definition 6.2 to generate $\mathsf{crs}'_{\mathscr{L}}$ as $(\mathsf{crs}'_{\mathscr{L}},\tau_\Sigma) \leftarrow \mathsf{TrapGen}'(\mathsf{par},\mathscr{L},\tau_{\mathscr{L}})$ instead of $\mathsf{crs}'_{\mathscr{L}} \leftarrow \mathsf{Gen}'_{\mathscr{L}}(\mathsf{par},\mathscr{L})$. We immediately have $|\Pr[W_1] - \Pr[W_0]| \leq \mathbf{Adv}^{\text{indist-}\Sigma}_{\mathscr{A}}(\lambda)$.

We note that the trapdoor τ_Σ produced by $\mathsf{TrapGen}'$ in Game_1 can be used in subsequent games to compute the $\mathsf{BadChallenge}$ function of the trapdoor Σ-protocol Π'.

Game$_2$: We modify the distribution of crs. Namely, at step 3 of $\mathsf{Gen}_{\mathscr{L}}$, we generate the keys for Π^{LPKE} as injective keys $(pk, sk) \leftarrow \mathsf{Keygen}(\Gamma)$ instead of lossy keys $(pk, sk) \leftarrow \mathsf{LKeygen}(\Gamma)$. The indistinguishability property of Π^{LPKE} guarantees $\Pr[W_2]$ is within negligible distance from $\Pr[W_1]$. We can easily build a distinguisher $\mathscr{B}$ against Π^{LPKE} such that

$$|\Pr[W_2] - \Pr[W_1]| \leq \mathbf{Adv}^{\mathsf{indist\text{-}LPKE}}_{\mathscr{B}}(\lambda).$$

Game$_3$: We introduce another change in the distribution of $\mathsf{crs}_{\mathscr{L}}$. We consider the relation R_{bad} defined by

$$\begin{aligned}\big((x,\mathbf{a}),\mathsf{Chall}\big) \in R_{\mathsf{bad}} \\ \Leftrightarrow\ \mathsf{Chall} = \mathsf{BadChallenge}\big(\tau_\Sigma, \mathsf{crs}'_{\mathscr{L}}, x, \mathsf{Decrypt}(sk,\mathbf{a})\big)\ .\end{aligned} \tag{6.7}$$

We now generate the key of the correlation-intractable hash function as $k \leftarrow \mathsf{StatGen}\big(1^\lambda, \mathsf{aux}_{R_{\mathsf{bad}}}\big)$ instead of $k \leftarrow \mathsf{Gen}(1^\lambda)$. Here, $\mathsf{aux}_{R_{\mathsf{bad}}}$ is the circuit that uses τ_Σ and sk to evaluate $\mathsf{BadChallenge}$ according to (6.7). The key indistinguishability property of $\mathscr{H}$ in the sense of Definition 6.7 implies $|\Pr[W_3] - \Pr[W_2]| \leq \mathbf{Adv}^{\text{indist-CI}}_{\mathscr{A}}(\lambda)$.

In Game_3, we claim that $\Pr[W_3] \leq 2^{-\Omega(\lambda)}$. Indeed, the statistical correlation intractability property of $\mathscr{H}$ implies that we can only have

$$\mathsf{Hash}(k,(x^\star,\mathbf{a}^\star)) = \mathsf{BadChallenge}\big(\tau_\Sigma, \mathsf{crs}'_{\mathscr{L}}, x^\star, \mathsf{Decrypt}(sk,\mathbf{a}^\star)\big)$$

with exponentially small probability. The probability to have $W_3 = \texttt{true}$ is thus smaller than $2^{-\Omega(\lambda)}$.

Putting the above altogether, the advantage of a PPT adversary is thus smaller than

$$\begin{aligned}\mathbf{Adv}^{\text{soundness}}_{\mathscr{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{indist\text{-}LPKE}}_{\mathscr{B}}(\lambda) \\ + \mathbf{Adv}^{\text{indist-}\Sigma}_{\mathscr{A}}(\lambda) + \mathbf{Adv}^{\text{indist-CI}}_{\mathscr{A}}(\lambda) + 2^{-\Omega(\lambda)},\end{aligned}$$

which proves the claim.

□

6.3.4 Application: Multi-Theorem Statistical NIZK for NP from LWE

By using the techniques from the previous section it is possible to get statistical NIZK for all NP languages, with non-adaptive soundness under the LWE assumption, in the Common *Random* String model. We describe the idea below.

The construction from the previous section can be applied to any trapdoor Σ-protocol. In particular, we can apply it to the modified FLS protocol of [CLW19], for the graph Hamiltonicity language, in which the fist prover message is already encrypted using a lossy scheme. For this reason we don't need to actually encrypt

it a second time, as a straight-forward application of our transformation would require. Instead, we can use the simulator of Theorem 6.2 and simulate proofs by equivocating lossy encryptions.

To encrypt the prover's first message in the FLS protocol, we can use the variant of Regev's LWE-based cryptosystem suggested in [GPV08, Section 8.1]. This is in fact a lossy encryption scheme with uniformly random lossy public keys. In the simulation, the uniformly random public matrix can be generated together with a trapdoor, thus enabling the equivocation of lossy ciphertexts.

Combining the above with the LWE-based somewhat Correlation-Intractable hash family proposed in [PS19] (that supports uniformly random hashing keys), we obtain a Multi-Theorem statistical NIZK argument for NP languages, with non-adaptive soundness under the LWE assumption, in the Common *Random* String model.

Multi-Theorem NIZK for NP can also be obtained generically using the FLS transformation from [FLS99]. The compiler works by combining a single-theorem NIZK and a PRG. In comparison with our result, their CRS is not sampled from a uniformly random distribution.

6.4 Direct Constructions of Unbounded Simulation-sound NIZK from Trapdoor Sigma-protocols

In this section we show how to upgrade the the construction from section 6.3.2 to obtain *unbounded simulation-soundness* (Definition 6.4) as well.

6.4.1 Overview

Continuing the discussion from section 6.3.1, we notice that lossy encryption with efficient opening is not enough to prove unbounded simulation soundness for our construction. The reason is that we need to be able to equivocate all the lossy ciphertexts in our simulation, while making sure that the proof the adversary outputs contains an injective (extractable) encryption, so we can prove simulation soundness.

This can be solved by using a lossy encryption scheme flavor, called R-lossy encryption with efficient opening (Definition 6.10). The encryption algorithm of such a scheme takes as input additional labels. Such a tag (label) can determine whether a ciphertext is lossy or injective, depending on the relation R. Now, when we use the injective mode of the encryption scheme to prove soundness, we are able to simulate the proofs by using lossy tags (on which we can equivocate the encryptions), while having an injective tag for the proof that the adversary outputs (thus making the encryption that the adversary outputs injective).

6.4.2 Direct Construction

Our proof systems is inspired by ideas from [GMY03, MY04, Gen04] and relies on the following ingredients:

- A trapdoor Σ-protocol $\Pi' = (\mathsf{Gen}'_{\mathsf{par}}, \mathsf{Gen}'_{\mathscr{L}}, \mathsf{P}', \mathsf{V}')$ with challenge space ChSp, for the gap language $\mathscr{L} = (\mathscr{L}_{\mathsf{zk}}, \mathscr{L}_{\mathsf{sound}})$ and which satisfies the properties of Definition 6.2. In addition, $\mathsf{BadChallenge}(\tau_\Sigma, \mathsf{crs}, x, \mathbf{a})$ should be computable in size $S \in \mathsf{poly}(\lambda)$ for any input $(\tau_\Sigma, \mathsf{crs}, x, \mathbf{a})$.

- A strongly unforgeable one-time signature scheme $\mathsf{OTS} = (\mathscr{G}, \mathscr{S}, \mathscr{V})$ with verification keys of length $\ell \in \mathsf{poly}(\lambda)$ (Definition 6.8).

- An admissible hash function $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$, for some $L \in \mathsf{poly}(\lambda)$ such that $L > \ell$, which induces the relation $R_{\mathsf{BM}} : \{0,1,\perp\}^L \times \{0,1\}^\ell \to \{0,1\}$ of Definition 6.11.

- An R-lossy PKE scheme given by the 7 algorithms:

  $\mathsf{R\text{-}LPKE} = (\mathsf{Par\text{-}Gen}, \mathsf{Keygen}, \mathsf{LKeygen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Opener}, \mathsf{Opener}')$ using the relation $R_{\mathsf{BM}} : \{0,1,\perp\}^L \times \{0,1\}^\ell \to \{0,1\}$ with public (resp. secret) key space $\mathscr{PK}$ (resp. $\mathscr{SK}$). We assume that the decryption algorithm Decrypt is computable in size S. We denote the message (resp. ciphertext) space by MsgSp (resp. CtSp) and the randomness space by R^{LPKE}. Let also D_R^{LPKE} denote the distribution from which the random coins of Encrypt are sampled.

- A somewhere correlation intractable hash family $\mathscr{H} = (\mathsf{Gen}, \mathsf{Hash})$ for the relation class $\mathscr{R}_{\mathsf{CI}}$ of relations that are efficiently searchable in size $2S$.

We also assume that these ingredients are compatible in the sense that P' outputs a first prover message $\mathbf{a}'$ that fits in the message space MsgSp of $\mathsf{R\text{-}LPKE}$. The correlation-intractable hash function should output values in the challenge space ChSp of the Σ-Protocol Π' as well.

The construction goes as follows.

$\mathsf{Gen}_{\mathsf{par}}(1^\lambda)$: Run $\mathsf{par} \leftarrow \mathsf{Gen}'_{\mathsf{par}}(1^\lambda)$ and output par.

$\mathsf{Gen}_{\mathscr{L}}(\mathsf{par}, \mathscr{L})$: Given public parameters par and a language $\mathscr{L} \subset \{0,1\}^N$, let $\mathscr{K} = \{0,1,\perp\}^L$ and $\mathscr{T} = \{0,1\}^\ell$. The CRS is generated as follows.

1. Generate a common reference string $\mathsf{crs}'_{\mathscr{L}} \leftarrow \mathsf{Gen}'_{\mathscr{L}}(\mathsf{par}, \mathscr{L})$ for the trapdoor Σ-protocol Π'.
2. Generate public parameters $\Gamma \leftarrow\!\!\!\!\!\rightarrow \mathsf{Par\text{-}Gen}(1^\lambda)$ for the R_{BM}-lossy PKE scheme where the relation $R_{\mathsf{BM}} : \mathscr{K} \times \mathscr{T} \to \{0,1\}$ is defined by an admissible hash function $\mathsf{AHF} : \{0,1\}^\ell \to \{0,1\}^L$. Choose a random value $K \leftarrow \mathscr{K}$ and generate lossy keys $(pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K)$.

3. Generate a key $k \leftarrow \mathsf{Gen}(1^\lambda)$ for a somewhere correlation intractable hash function with output in the challenge space ChSp of the Σ-Protocol Π'.

Output the language-dependent $\mathsf{crs}_{\mathscr{L}} := \left(\mathsf{crs}'_{\mathscr{L}}, k\right)$ and the simulation trapdoor $\tau_{\mathsf{zk}} := sk$, which is the lossy secret key of R-LPKE. The global common reference string consists of $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathscr{L}}, pk, \mathsf{AHF}, \Pi^{\mathrm{ots}})$.

P(crs, x, w) : To prove a statement x using a witness $w \in R_{\mathsf{zk}}(x)$, generate a one-time signature key pair $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathscr{G}(1^\lambda)$. Then, do the following.

1. Compute $(\mathbf{a}', st') \leftarrow \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}}, x, w)$ by invoking the prover for Π'. Then compute $\mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathsf{VK}, \mathbf{a}'; \mathbf{r})$ using random coins $\mathbf{r} \hookleftarrow D_R^{\mathsf{LPKE}}$ that are sampled from the distribution D_R^{LPKE} over R^{LPKE}.
2. Compute $\mathsf{Chall} := \mathsf{Hash}(k, (x, \mathbf{a}, \mathsf{VK})) \in \mathsf{ChSp}$.
3. Compute $\mathbf{z}' = \mathsf{P}'(\mathsf{crs}'_{\mathscr{L}}, x, w, \mathbf{a}', \mathsf{Chall}, st')$. Define $\mathbf{z} = (\mathbf{a}', \mathbf{z}', \mathbf{r})$.
4. Generate a one-time signature $sig \leftarrow \mathscr{S}(\mathsf{SK}, (x, \mathbf{a}, \mathbf{z}))$ and output the proof $\boldsymbol{\pi} = \left(\mathsf{VK}, (\mathbf{a}, \mathbf{z}), sig\right)$.

V$(\mathsf{crs}, x, \boldsymbol{\pi})$: Given a statement x, and a purported proof $\boldsymbol{\pi} = \left(\mathsf{VK}, (\mathbf{a}, \mathbf{z}), sig\right)$, return 0 if $\mathscr{V}(\mathsf{VK}, (x, \mathbf{a}, \mathbf{z}), sig) = 0$. Otherwise,

1. Write $\mathbf{z}$ as $\mathbf{z} = (\mathbf{a}', \mathbf{z}', \mathbf{r})$ and return 0 if it does not parse properly. Return 0 if $\mathbf{a} \neq \mathsf{Encrypt}(pk, \mathsf{VK}, \mathbf{a}'; \mathbf{r})$ or $\mathbf{r} \notin R^{\mathsf{LPKE}}$.
2. Let $\mathsf{Chall} = \mathsf{Hash}\left(k, (x, (\mathbf{a}, \mathsf{VK})\right)$. If $\mathsf{V}'(\mathsf{crs}'_{\mathscr{L}}, x, (\mathbf{a}', \mathsf{Chall}, \mathbf{z}')) = 1$ then return 1. Otherwise, return 0.

The NIZK simulator is very similar to the one used in the proof of Theorem 6.2 and the proof follows a similar pattern.

Theorem 6.4. *Assuming that the trapdoor Σ-protocol Π' is (statistical) special zero-knowledge, the above argument system is (statistical) multi-theorem zero-knowledge*

Sketch. We describe a simulator $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ which uses the lossy secret key $\tau_{\mathsf{zk}} = sk$ of R-LPKE to simulate proofs without using the witnesses. Namely, on input of $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{par}}(1^\lambda)$, Sim_0 generates $\mathsf{crs}_{\mathscr{L}}$ by proceeding identically to $\mathsf{Gen}_{\mathscr{L}}$ while Sim_1 is described hereunder.

Sim$_1(\mathsf{crs}, \tau_{\mathsf{zk}}, x)$**:** On input a statement $x \in \{0,1\}^N$ and the simulation trapdoor $\tau_{\mathsf{zk}} = sk$, algorithm Sim_1 proceeds as follows.

1. Generate a one-time signature key pair $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathscr{G}(1^\lambda)$. Let $\mathbf{0}^{|\mathbf{a}'|}$ the all-zeroes string of length $|\mathbf{a}'|$. Sample random coins $\mathbf{r}_0 \hookleftarrow D_R^{\mathsf{LPKE}}$ from the distribution D_R^{LPKE} and compute $\mathbf{a} \leftarrow \mathsf{Encrypt}(pk, \mathsf{VK}, \mathbf{0}^{|\mathbf{a}'|}; \mathbf{r}_0)$.

2. Compute $\mathsf{Chall} = \mathsf{Hash}(k, (x, \mathbf{a}, \mathsf{VK}))$.
3. Run the special ZK simulator $(\mathbf{a}', \mathbf{z}') \leftarrow \mathsf{ZKSim}(\mathsf{crs}'_{\mathscr{L}}, x, \mathsf{Chall})$ of Π' so as to obtain a simulated transcript $(\mathbf{a}', \mathsf{Chall}, \mathbf{z}')$ of Π' for the challenge Chall.
4. Using the lossy secret key sk of R-LPKE, compute random coins $\mathbf{r} \leftarrow \mathsf{Opener}'(sk, \mathsf{VK}, \mathbf{a}, \mathbf{a}')$ which explain $\mathbf{a}$ as an encryption of $\mathbf{a}'$ under the tag VK. Then, define $\mathbf{z} = (\mathbf{a}', \mathbf{z}', \mathbf{r})$
5. Generate a one-time signature $sig \leftarrow \mathscr{S}\big(\mathsf{SK}, (x, \mathbf{a}, \mathbf{z})\big)$ and output the proof $\boldsymbol{\pi} = \big(\mathsf{VK}, (\mathbf{a}, \mathbf{z}), sig\big)$.

The proof proceeds in the same fashion as the proof of Theorem 6.2, so the details are omitted here. □

6.4.3 Proof of Unbounded Simulation-Soundness

Theorem 6.5. *The above argument system is unbounded simulation-sound if: (i)* OTS *is a strongly unforgeable one-time signature; (ii)* R-LPKE *is an R_{BM}-lossy PKE scheme; (iii) The hash family $\mathscr{H}$ is somewhere correlation-intractable for all relations that are searchable in size $2S$, where S denotes the maximal running time of algorithms* $\mathsf{BadChallenge}(\cdot,\cdot,\cdot,\cdot)$ *and* $\mathsf{Decrypt}(\cdot,\cdot,\cdot)$.

Proof. We consider a sequence of games where, for each i, we define a variable $W_i \in \{\mathtt{true}, \mathtt{false}\}$ where $W_i = \mathtt{true}$ if and only if the adversary wins in Game_i.

Game_0: This is the real game of Definition 6.4. Namely, the challenger first runs $(\mathsf{crs}, \tau_{\mathrm{zk}}) \leftarrow \mathsf{Sim}_0(\mathsf{par}, 1^N)$ and gives $\mathsf{crs} = (\mathsf{par}, \mathsf{crs}_{\mathscr{L}}, pk, \mathsf{AHF}, \Pi^{\mathrm{ots}})$ to the adversary $\mathscr{A}$. At the same time, the challenger generates a trapdoor $\tau_{\mathscr{L}}$ for the language $\mathscr{L}_{\mathsf{sound}}$ in such a way that it can efficiently test if $\mathscr{A}$'s output satisfies $x^\star \notin \mathscr{L}_{\mathsf{sound}}$. The adversary is granted oracle access to $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot, \cdot)$. At each query, $\mathscr{A}$ chooses a statement $x \in \{0,1\}^N$ and the challenger replies by returning a simulated argument $\boldsymbol{\pi} \leftarrow \mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, x)$. When $\mathscr{A}$ halts, it outputs a pair $(x^\star, \boldsymbol{\pi}^\star)$, where $\boldsymbol{\pi}^\star = \big(\mathsf{VK}^\star, (\mathbf{a}^\star, \mathbf{z}^\star), sig^\star\big)$. The Boolean variable W_0 is thus set to $W_0 = \mathtt{true}$ under the following three conditions: (i) $(x^\star, \boldsymbol{\pi}^\star) \notin \mathscr{Q}$, where $\mathscr{Q} = \{(x_i, \boldsymbol{\pi}_i)\}_{i=1}^{Q}$ is the set of queries to $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot, \cdot)$ and the corresponding responses $\boldsymbol{\pi}_i = \big(\mathsf{VK}^{(i)}, (\mathbf{a}_i, \mathbf{z}_i), sig_i\big)$; (ii) $x^\star \notin \mathscr{L}_{\mathsf{sound}}$; and (iii) $V(\mathsf{crs}, x^\star, \boldsymbol{\pi}^\star) = 1$. We may assume w.l.o.g. that the one-time verification keys $\{\mathsf{VK}^{(i)}\}_{i=1}^{Q}$ are chosen ahead of time at the beginning of the game. By definition we have $\mathbf{Adv}^{\mathrm{uss}}_{\mathscr{A}}(\lambda) = \Pr[W_0]$.

Game_1: This is like Game_0 except that the challenger $\mathscr{B}$ sets $W_1 = \mathtt{false}$ if $\mathscr{A}$ outputs a fake proof $(x^\star, \boldsymbol{\pi}^\star)$, where $\boldsymbol{\pi}^\star = \big(\mathsf{VK}^\star, (\mathbf{a}^\star, \mathbf{z}^\star), sig^\star\big)$ contains a $\mathsf{VK}^\star$ that coincides with the verification key $\mathsf{VK}^{(i)}$ contained in an output $\boldsymbol{\pi}_i = \big(\mathsf{VK}^{(i)}, (\mathbf{a}_i, \mathbf{z}_i), sig_i\big)$ of $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot, \cdot)$. The strong unforgeability of

OTS implies that $\Pr[W_1]$ cannot noticeably differ from $\Pr[W_0]$. We can easily turn $\mathscr{B}$ into a forger such that $|\Pr[W_1] - \Pr[W_0]| \leq \mathbf{Adv}^{\mathrm{ots}}_{\mathscr{B}}(\lambda)$.

Game_2: This game is like Game_1 with the following changes. At step 2 of $\mathsf{Gen}_{\mathscr{L}}$, the challenger runs $K \leftarrow \mathsf{AdmSmp}(1^\lambda, Q, \delta)$ to generate a key $K \in \{0,1,\perp\}^L$ for an admissible hash function $\mathsf{AHF} : \{0,1\}^\ell \rightarrow \{0,1\}^L$, where Q is an upper bound on the number of adversarial queries. By the second indistinguishability property of the R_{BM}-lossy PKE scheme, we know that changing the initialization value does not significantly affect $\mathscr{A}$'s view. It follows that

$$|\Pr[W_2] - \Pr[W_1]| \leq \mathbf{Adv}^{\mathsf{indist\text{-}LPKE\text{-}2}}_{\mathscr{B}}(\lambda).$$

Game_3: This game is identical to Game_2 with one modification. When the adversary halts and outputs $x^\star$, the challenger checks if the conditions

$$F_{\mathsf{ADH}}(K, \mathsf{VK}^{(1)}) = \cdots = F_{\mathsf{ADH}}(K, \mathsf{VK}^{(Q)}) = 1 \ \wedge \ F_{\mathsf{ADH}}(K, \mathsf{VK}^\star) = 0 \tag{6.8}$$

are satisfied, where $\mathsf{VK}^\star$ is the one-time verification key in the adversary's output and $\mathsf{VK}^{(1)}, \ldots, \mathsf{VK}^{(Q)}$ are those in adversarial queries. If these conditions do not hold, the challenger aborts and sets $W_3 = \texttt{false}$. For simplicity, we assume that $\mathscr{B}$ aborts at the very beginning of the game if it detects that there exists $i \in [Q]$ such that $F_{\mathsf{ADH}}(K, \mathsf{VK}^{(i)}) = 0$ (recall that $\{\mathsf{VK}^{(i)}\}_{i=1}^{Q}$ are chosen at the outset of the game by $\mathscr{B}$). If conditions (6.8) are satisfied, the challenger sets $W_3 = \texttt{true}$ whenever $W_1 = \texttt{true}$. Letting Fail denote the event that $\mathscr{B}$ aborts because (6.8) does not hold, we have $W_3 = W_2 \wedge \neg\mathsf{Fail}$. Since the key K of the admissible hash function is statistically independent of the adversary's view, we can apply Theorem 2.17 to argue that there is a noticeable function $\delta(\lambda)$ such that $\Pr[\neg\mathsf{Fail}] \geq \delta(\lambda)$. This implies

$$\Pr[W_3] \quad = \quad \Pr[W_2 \wedge \neg\mathsf{Fail}] \geq \delta(\lambda) \cdot \Pr[W_2] \ , \tag{6.9}$$

where the inequality stems from the fact that Fail is independent of W_1 since K is statistically independent of $\mathscr{A}$'s view.

We remark that, if conditions (6.8) are satisfied in Game_2, the of one-time verification keys $(\mathsf{VK}^{(1)}, \ldots, \mathsf{VK}^{(Q)}, \mathsf{VK}^\star)$ satisfy $R_{\mathsf{BM}}(K, \mathsf{VK}^\star) = 1$ and $R_{\mathsf{BM}}(K, \mathsf{VK}^{(i)}) = 0$ for all $i \in [Q]$.

Game_4: In this game, we modify the oracle $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot, \cdot)$ and by exploiting the efficient opening property of R-LPKE for lossy tags (instead of lossy keys). At the i-th query x_i to $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot, \cdot)$, we must have $F_{\mathsf{ADH}}(K, \mathsf{VK}^{(i)}) = 1$ (meaning that $\mathsf{VK}^{(i)}$ is a lossy tag as $R_{\mathsf{BM}}(K, \mathsf{VK}^{(i)}) = 0$) if $\mathscr{B}$ did not abort. This allows $\mathscr{B}$ to equivocate $\mathbf{a}$ using the trapdoor key tk instead of the lossy secret key sk of R-LPKE. Namely, at step 4 of Sim_1, the modified oracle computes random coins $\mathbf{r} \leftarrow \mathsf{Opener}(pk, tk, \mathsf{VK}, \mathbf{a}, \mathbf{a}')$ instead of running Opener'

using sk. We define the Boolean variable W_4 exactly as W_3. Since Opener and Opener$'$ output samples from the same distribution D_R^{LPKE} over R^{LPKE}, this implies that $|\Pr[W_4] - \Pr[W_3]| \leq 2^{-\lambda}$.

Game$_5$: We now modify the distribution of crs. At step 2 of Gen, we generate the keys for R-LPKE as injective keys $(pk, sk, tk) \leftarrow \mathsf{Keygen}(\Gamma, K)$ instead of lossy keys $(pk, sk, tk) \leftarrow \mathsf{LKeygen}(\Gamma, K)$. The indistinguishability property (i) of R-LPKE ensures that $\Pr[W_5]$ and $\Pr[W_4]$ are negligibly far apart. Recall that this indistinguishability property ensures that the distributions of pairs (pk, tk) produced by Keygen and LKeygen are computationally indistinguishable. So we can easily build a distinguisher $\mathscr{B}$ against R-LPKE that bridges between Game$_4$ and Game$_5$ (by using tk to simulate $\mathsf{Sim}_1(\mathsf{crs}, \tau_{\mathrm{zk}}, \cdot, \cdot)$ as in Game$_4$). It comes that $|\Pr[W_5] - \Pr[W_4]| \leq \mathbf{Adv}^{\text{indist-LPKE-1}}_{\mathscr{B}}(\lambda)$.

Due to the modification introduced in Game$_5$, if the conditions (6.8) are satisfied, we have $R_{\mathsf{BM}}(K, \mathsf{VK}^\star) = 1$, meaning that the adversary's fake proof $\boldsymbol{\pi}^\star = \big(\mathsf{VK}^\star, \big(\mathbf{a}^\star, \mathbf{z}^\star = (\mathbf{a}'^\star, \mathbf{z}'^\star, \mathbf{r}^\star)\big), sig^\star\big)$ involves an injective tag $\mathsf{VK}^\star$. Since pk is now an injective key, this implies that $\mathbf{a}^\star$ is an injective encryption of $\mathbf{a}'^\star$ under the tag $\mathsf{VK}^\star$ using the randomness $\mathbf{r}^\star$.

Game$_6$: We introduce another change in the distribution of $\mathsf{crs}_{\mathscr{L}}$. We consider the relation R_{bad} defined by

$$\begin{aligned} &((x, \mathbf{a}, \mathsf{VK}), \mathsf{Chall}) \in R_{\mathsf{bad}} \\ &\quad \Leftrightarrow \ \mathsf{Chall} = \mathsf{BadChallenge}(\tau_\Sigma, \mathsf{crs}'_{\mathscr{L}}, x, \mathsf{Decrypt}(sk, \mathsf{VK}, \mathbf{a})). \end{aligned} \tag{6.10}$$

We now generate the key of the correlation-intractable hash function as $k \leftarrow \mathsf{StatGen}\big(1^\lambda, \mathsf{aux}_{R_{\mathsf{bad}}}\big)$ instead of $k \leftarrow \mathsf{Gen}(1^\lambda)$. Here, $\mathsf{aux}_{R_{\mathsf{bad}}}$ is the circuit that evaluates BadChallenge as per (6.10), where τ_Σ and sk are hard-wired. Here, we crucially rely on the fact that Definition 6.7 captures key indisitinguishability even when aux_R is given to the adversary. (This is necessary since aux_R depends on τ_Σ and $\tau_{\mathscr{L}}$ and the latter is given to $\mathscr{A}$). By this key indistinguishability property of $\mathscr{H}$, we have $|\Pr[W_6] - \Pr[W_5]| \leq \mathbf{Adv}^{\text{indist-CI}}_{\mathscr{A}}(\lambda)$.

In Game$_6$, we claim that $\Pr[W_6] \leq 2^{-\Omega(\lambda)}$. Indeed, if $\mathscr{B}$ did not fail, we know that the adversary's output $\boldsymbol{\pi}^\star = \big(\mathsf{VK}^\star, \big(\mathbf{a}^\star, \mathbf{z}^\star = (\mathbf{a}'^\star, \mathbf{z}'^\star, \mathbf{r}^\star)\big), sig^\star\big)$ involves an injective tag $\mathsf{VK}^\star$, so that $\mathbf{a}^\star$ is a statistically binding commitment to $\mathbf{a}'^\star$. So there exists only one message $\mathbf{a}'^\star$ such that $\mathbf{a}^\star = \mathsf{Encrypt}(pk, \mathsf{VK}^\star, \mathbf{a}'^\star; \mathbf{r}^\star)$ for some $\mathbf{r}^\star \in R^{\mathsf{LPKE}}$. Said otherwise, there exists only one $\mathbf{a}'^\star$ for which a pair $(\mathbf{a}'^\star, \mathbf{r}^\star)$ satisfies step 1 of the verification algorithm. Moreover, since $\mathbf{a}^\star$ uniquely determines $\mathbf{a}'^\star$, the statistical correlation intractability property of $\mathscr{H}$ implies that we can only have

$$\mathsf{Hash}(k,(x^\star,\mathbf{a}^\star,\mathsf{VK}^\star)) = \mathsf{BadChallenge}\big(\tau_\Sigma,\mathsf{crs}'_{\mathscr{L}},x^\star,\mathsf{Decrypt}(sk,\mathsf{VK}^\star,\mathbf{a}^\star)\big)$$

with exponentially small probability. The probability to have $W_6 = \texttt{true}$ is thus smaller than $2^{-\Omega(\lambda)}$ as claimed.

Putting the above altogether, we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathrm{uss}}_{\mathscr{A}}(\lambda) \leq \mathbf{Adv}^{\mathrm{ots}}_{\mathscr{B}}(\lambda) \quad &+ \quad \mathbf{Adv}^{\text{indist-LPKE-2}}_{\mathscr{B}}(\lambda) + \frac{1}{\delta(\lambda)}\cdot\Big(\mathbf{Adv}^{\text{indist-LPKE-1}}_{\mathscr{B}}(\lambda)\\ &+ \quad \mathbf{Adv}^{\text{indist-}\Sigma}_{\mathscr{B}}(\lambda) + \mathbf{Adv}^{\text{indist-CI}}_{\mathscr{B}}(\lambda) + 2^{-\Omega(\lambda)}\Big)\ ,\end{aligned}$$

which completes the proof.

□

We note that the above security proof is not tight as the use of admissible hash functions induces a security loss $1/\delta(\lambda)$ (where $\delta(\lambda)$ is the non-negligible function of Theorem 2.16) in the upper bound on the adversary's advantage.

In [LNPT19] it was actually shown that it is possible to to have a tightly secure unbounded-simulation sound NIZK scheme by constructing and using an R-lossy PKE corresponding to a relation induced by a PRF: $\mathscr{K}\times\{0,1\}^\ell\to\{0,1\}^\lambda$, instead of the AHF. The relation $R_{\mathsf{PRF}}:\mathscr{K}\times\{0,1\}^\ell\to\{0,1\}$ is defined as $R_{\mathsf{PRF}}(K,(t_a,t_c)) = 1$ iff $t_c \neq \mathsf{PRF}(K,t_a)$.

6.4.4 Application: Naor-Yung KDM-CCA2 scheme from LWE

In this section we discuss how to apply the Naor-Yung paradigm and the NIZK construction from section 6.4.2 to obtain the most efficient public-key encryption scheme, providing key-dependent message security under adaptive chosen-ciphertext attacks (or KDM-CCA2 security for short), under the standard Learning-With-Errors (LWE) assumption.

Naor-Yung Transform

Naor and Yung [NY90] showed how to use NIZK schemes to obtain public-key encryption, secure against chosen-ciphertext (CCA1) attacks ('lunchtime' attacks), from any scheme that is secure against chosen-plaintext attacks (CPA). They encrypted the same message twice, using two different public keys, and then computed a non-interactive zero-knowledge proof that the encryptions are of the same message. Sahai later showed [Sah99] that the double encryption paradigm gives CCA2 security if the NIZK scheme satisfies the simulation-soundness property.

Key Dependent Message (KDM) Security

A public-key encryption with Key-Dependent Message (KDM) security [BRS02], assures privacy against adversaries that are able to obtain encryption of messages that depend on the secret key of the scheme. This security notion is useful in situations that may appear due to careless key management or when using disk encryption utilities [BHHO08]. It is also useful for the bootstrapping step in Fully Homomorphic Encryptions [Gen09].

Constructions of public-key KDM-CCA2 secure schemes under lattice assumptions, were previously known to be possible from any KDM-CPA scheme, by combining the generic NIZK techniques [FLS99, SCO+01] and the recent results of [CLW19, PS19] with the Naor-Yung [NY90] transform. Unfortunately such schemes are rather inefficient, as the NIZK construction goes through a Karp reduction to the graph Hamiltonicity problem.

To obtain more efficient KDM-CCA2 schemes, we apply the the Naor-Yung paradigm to the scheme of Applebaum, Cash, Peikert, Sahai [ACPS09], which was proved secure under the LWE assumption, but only against key-dependent message for chosen-plaintext attacks (KDM-CPA). In order to prove the stronger KDM-CCA2 security, we use an unbounded simulation-sound NIZK that proves that two ACPS ciphertexts encrypt the same message. To this end, we first construct a trapdoor Σ-protocol for proving that two ACPS ciphertexts encrypt the same message (see Section 6.4.5), then we apply the generic transformation from section 6.4.2. By using this trapdoor Σ-protocol and our generic compiler, we obtain an unbounded simulation-sound NIZK, without going through the Karp reduction. Thus we obtain a more efficient public-key encryption scheme in the end.

The details and the proof of the KDM-CCA2 construction can be checked in [LNPT19], but they follow the same template as any Naor-Yung CCA2 scheme.

6.4.5 A Trapdoor Σ-Protocol for ACPS Ciphertexts

The KDM-CPA system of Applebaum *et al.* [ACPS09] uses a modulus $q = p^2$, for some prime p. Its public key $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ contains a random matrix $\mathbf{A} \sim U(\mathbb{Z}_q^{n\times m})$ and a vector $\mathbf{b} = \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}$, for some $\mathbf{s} \sim D_{\mathbb{Z}^n,\alpha q}$, $\mathbf{e} \sim D_{\mathbb{Z}^m,\alpha q}$. Its encryption algorithm proceeds analogously to the primal Regev cryptosystem [Reg05] and computes $\mathbf{c} = (\bar{\mathbf{c}}, c) = (\mathbf{A}\cdot\mathbf{r}, \mathbf{b}^\top\mathbf{r} + \mu\cdot p + \chi) \in \mathbb{Z}_q^{n+1}$, where $\mathbf{r} \sim D_{\mathbb{Z}^m,r}$ is a Gaussian vector and $\chi \sim D_{\mathbb{Z},r'}$ is sampled from a Gaussian with a slightly larger standard deviation. Decryption proceeds by rounding $c - \mathbf{s}^\top \cdot \bar{\mathbf{c}} \mod q$ to the nearest multiple of p.

In this section, we describe a trapdoor Σ-protocol allowing to prove that two ACPS ciphertexts $\mathbf{c}_0 = (\bar{\mathbf{c}}_0, c_0)$, $\mathbf{c}_1 = (\bar{\mathbf{c}}_1, c_1)$ are both encryptions of the same $\mu \in \mathbb{Z}_p$.

PROVING PLAINTEXT EQUALITIES IN ACPS CIPHERTEXTS. Let $q = p^2$, for some prime p, and a matrix $\mathbf{A}$ which is used to set up two Regev public keys $(\mathbf{A}, \mathbf{b}_0) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ and $(\mathbf{A}, \mathbf{b}_1) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, where $\mathbf{b}_0 = \mathbf{A}^\top \cdot \mathbf{s}_0 + \mathbf{e}_0$ and $\mathbf{b}_1 = \mathbf{A}^\top \cdot \mathbf{s}_1 + \mathbf{e}_1$ for

some $\mathbf{s}_0, \mathbf{s}_1 \sim D_{\mathbb{Z}^n,\alpha q}$, $\mathbf{e}_0, \mathbf{e}_1 \sim D_{\mathbb{Z}^m,\alpha q}$. Let also the matrix

$$\mathbf{A}_{\mathrm{eq}} = \left[\begin{array}{c|c|c|c} \mathbf{A} & & & \\ \hline \mathbf{b}_0^\top & 1 & & \\ \hline & & \mathbf{A} & \\ \hline & & \mathbf{b}_1^\top & 1 \end{array}\right] \in \mathbb{Z}_q^{2(n+1)\times 2(m+1)}, \tag{6.11}$$

We give a trapdoor Σ-protocol for the language $\mathscr{L}^{\mathrm{eq}} = (\mathscr{L}^{\mathrm{eq}}_{\mathsf{zk}}, \mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}})$, where

$$\begin{aligned} \mathscr{L}^{\mathrm{eq}}_{\mathsf{zk}} \quad := \quad & \Big\{(\mathbf{c}_0,\mathbf{c}_1) \in (\mathbb{Z}_q^{n+1})^2 \mid \exists \mathbf{r}_0, \mathbf{r}_1 \in \mathbb{Z}^m,\ \chi_0, \chi_1 \in \mathbb{Z},\ \mu \in \mathbb{Z}_p \ : \\ & \qquad \|\mathbf{r}_b\| \le B_r,\ |\chi_b| \le B_\chi \quad \forall b \in \{0,1\} \\ & \qquad \wedge \quad \mathbf{c}_b = \bar{\mathbf{A}}_b \cdot [\mathbf{r}_b^\top \mid \chi_b]^\top + \mu \cdot \left[\mathbf{0}^{n\top} \mid p\right]^\top \bmod q \Big\}, \\ \mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}} \quad := \quad & \Big\{(\mathbf{c}_0,\mathbf{c}_1) \in (\mathbb{Z}_q^{n+1})^2 \mid \exists\, \bar{\mathbf{c}}_0, \bar{\mathbf{c}}_1 \in \mathbb{Z}_q^n,\ v_0, v_1 \in [-B^*, B^*],\ \mu \in \mathbb{Z}_p \\ & \qquad \wedge \quad \mathbf{c}_b = \left[\begin{array}{c} \bar{\mathbf{c}}_b \\ \hline \mathbf{s}_b^\top \cdot \bar{\mathbf{c}}_b + p \cdot \mu + v_b \end{array}\right] \quad \forall b \in \{0,1\} \Big\}, \end{aligned}$$

where

$$\bar{\mathbf{A}}_b = \left[\begin{array}{c|c} \mathbf{A} & \\ \hline \mathbf{b}_b^\top & 1 \end{array}\right] \in \mathbb{Z}_q^{(n+1)\times(m+1)} \qquad \forall b \in \{0,1\}.$$

We note that $\mathscr{L}^{\mathrm{eq}}_{\mathsf{zk}} \subseteq \mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}}$ when $B_r \alpha q \sqrt{m} + B_\chi < B^* \ll p$. Also, $\mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}}$ is equivalently defined as the language of pairs $(\mathbf{c}_0, \mathbf{c}_1)$ such that such that

$$\left[\begin{array}{c|c|c|c} -\mathbf{s}_0^\top & 1 & & \\ \hline & & -\mathbf{s}_1^\top & 1 \end{array}\right] \cdot \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \end{bmatrix} \bmod q \ = \ \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} + \mu \cdot \begin{bmatrix} p \\ p \end{bmatrix}$$

for some $\mu \in \mathbb{Z}_p$, $v_0, v_1 \in [-B^*, B^*]$.

$\mathsf{Gen}_{\mathsf{par}}(1^\lambda)$: On input of a security parameter $\lambda \in \mathbb{N}$, choose moduli q, p with $q = p^2$, dimensions n, m, and error rate $\alpha > 0$ and a Gaussian parameter $\sigma_{\mathrm{eq}} \ge \log(2m+2) \cdot \sqrt{B_r^2 + B_\chi^2}$. Define public parameters $\mathsf{par} = \{\lambda, q, p, n, m, \alpha, \sigma_{\mathrm{eq}}\}$.

$\mathsf{Gen}_{\mathscr{L}}(\mathsf{par}, \mathscr{L}^{\mathrm{eq}})$: Takes in global parameters par and the description of a language $\mathscr{L}^{\mathrm{eq}} = (\mathscr{L}^{\mathrm{eq}}_{\mathsf{zk}}, \mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}})$ specifying real numbers $B^*, B_r, B_\chi > 0$ such that $B_r \alpha q \sqrt{m} + B_\chi < B^* \ll p$, and a matrix $\mathbf{A}_{\mathrm{eq}}$ from the distribution (6.11). It defines the language-dependent $\mathsf{crs}_{\mathscr{L}} = \{\bar{\mathbf{A}}, B^*, B_r, B_\chi\}$. The global CRS is

$$\mathsf{crs} = \big(\{\lambda, q, p, n, m, \alpha, \sigma_{\mathrm{eq}}\}, \{\mathbf{A}_{\mathrm{eq}}, B^*, B_r, B_\chi\}\big).$$

$\mathsf{TrapGen}(\mathsf{par}, \mathscr{L}, \tau_{\mathscr{L}})$: Given par and a language description $\mathscr{L}^{\mathrm{eq}}$ that specifies $B^*, B_r, B_\chi > 0$ satisfying the same constraints as in $\mathsf{Gen}_{\mathscr{L}}$, a matrix $\mathbf{A}_{\mathrm{eq}}$ sampled from the distribution (6.11), as well as a membership-testing trapdoor $\tau_{\mathscr{L}} = (\mathbf{s}_0, \mathbf{s}_1) \sim (D_{\mathbb{Z}^n,\alpha q})^2$ for $\mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}}$, output $\mathsf{crs}_{\mathscr{L}} = \{\bar{\mathbf{A}}, B^*, B_r, B_\chi\}$. The global CRS is $\mathsf{crs} = \big(\{\lambda, q, p, n, m, \alpha, \sigma_{\mathrm{eq}}\}, \{\mathbf{A}_{\mathrm{eq}}, B^*, B_r, B_\chi\}\big)$ and the trapdoor $\tau_\Sigma = (\mathbf{s}_0, \mathbf{s}_1) \in \mathbb{Z}^n \times \mathbb{Z}^n$.

$\mathbf{P}\big(\mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), (\mu, \mathbf{w})\big) \leftrightarrow \mathbf{V}(\mathsf{crs}, \mathbf{x})$: Given crs and a statement

$$\left[\frac{\mathbf{c}_0}{\mathbf{c}_1}\right] = \mathbf{A}_{\mathrm{eq}} \cdot [\mathbf{r}_0{}^\top \mid \chi_0 \mid \mathbf{r}_1{}^\top \mid \chi_1]^\top + \mu \cdot [\mathbf{0}^{n\top} \mid p \mid \mathbf{0}^{n\top} \mid p]^\top \in \mathbb{Z}_q^{2(n+1)},$$

the prover P (who has $\mathbf{w} = [\mathbf{r}_0^\top \mid \chi_0 \mid \mathbf{r}_1^\top \mid \chi_1]^\top \in \mathbb{Z}^{2(m+1)}$ and $\mu \in \mathbb{Z}_p$) and the verifier V interact as follows.

1. The prover P samples a uniform scalar $r_\mu \hookleftarrow U(\mathbb{Z}_p)$ and Gaussian vector $\mathbf{r}_w \hookleftarrow D_{\mathbb{Z}^{2(m+1)}, \sigma_{\mathrm{eq}}}$. It computes the following which is sent to V:
$$\mathbf{a} = \mathbf{A}_{\mathrm{eq}} \cdot \mathbf{r}_w + r_\mu \cdot [\mathbf{0}^{n\top} \mid p \mid \mathbf{0}^{n\top} \mid p]^\top \in \mathbb{Z}_q^{2(n+1)}.$$
2. V sends a random challenge $\mathsf{Chall} \in \{0,1\}$ to P.
3. P computes $\mathbf{z} = \mathbf{r}_w + \mathsf{Chall} \cdot \mathbf{w} \in \mathbb{Z}^{2(m+1)}$, $z_\mu = r_\mu + \mathsf{Chall} \cdot \mu \mod p$. It sends $(\mathbf{z}, z_\mu)$ to V with probability $\theta = \min\Big(\frac{D_{\mathbb{Z}^{2(m+1)}, \sigma_{\mathrm{eq}}}(\mathbf{z})}{M \cdot D_{\mathbb{Z}^{2(m+1)}, \sigma_{\mathrm{eq}}, \mathsf{Chall}\cdot\mathbf{w}}(\mathbf{z})}, 1\Big)$, where $M = e^{12/\log(2(m+1)) + 1/(2\log^2(2(m+1)))}$. P aborts, with probability $1-\theta$.
4. Given $(\mathbf{z}, z_\mu) \in \mathbb{Z}^{2(m+1)} \times \mathbb{Z}_p$, V checks if $\|\mathbf{z}\| \le \sigma_{\mathrm{eq}} \sqrt{2(m+1)}$ and
$$\mathbf{a} + \mathsf{Chall} \cdot \left[\frac{\mathbf{c}_0}{\mathbf{c}_1}\right] = \mathbf{A}_{\mathrm{eq}} \cdot \mathbf{z} + z_\mu \cdot [\mathbf{0}^{n\top} \mid p \mid \mathbf{0}^{n\top} \mid p]^\top \bmod q. \tag{6.12}$$
If these conditions do not both hold, V halts and returns $\perp$.

$\mathbf{BadChallenge}\big(\mathsf{par}, \tau_\Sigma, \mathsf{crs}, (\mathbf{c}_0, \mathbf{c}_1), \mathbf{a}\big)$: Given $\tau_\Sigma = (\mathbf{s}_0, \mathbf{s}_1) \in \mathbb{Z}^n \times \mathbb{Z}^n$, parse the first prover message as $\mathbf{a} = (\mathbf{a}_0^\top \mid \mathbf{a}_1^\top)^\top \in \mathbb{Z}_q^{2(n+1)}$. If there exists $d \in \{0,1\}$ such that no pair $(\mu'_d, \mathbf{v}_d) \in [-(p-1)/2, (p-1)/2] \times [-B^*/2, B^*/2]^2$ satisfies

$$\left[\begin{array}{c|c|c|c} -\mathbf{s}_0^\top & 1 & & \\ \hline & & -\mathbf{s}_1^\top & 1 \end{array}\right] \cdot (\mathbf{a} + d \cdot \mathbf{c}) \bmod q = \mathbf{v}_d + \mu'_d \cdot \left[\frac{p}{p}\right] \tag{6.13}$$

over $\mathbb{Z}$, then return $\mathsf{Chall} = 1 - d$. Otherwise, return $\mathsf{Chall} = \perp$.

The completeness of the protocol crucially uses the fact that p divides q to ensure that the response $z_\mu = r_\mu + \mathsf{Chall} \cdot \mu \bmod p$ satisfies (6.12).

The intuition of BadChallenge is that, for a false statement $(\mathbf{c}_0, \mathbf{c}_1) \notin \mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}}$, there exists $d \in \{0,1\}$ such that no pair $(\mu'_d, \mathbf{v}_d)$ satisfies (6.13) for a small enough $\mathbf{v}_d \in \mathbb{Z}^2$. Moreover, for this challenge $\mathsf{Chall} = d$, no valid response can exist, as shown in the proof of Lemma 6.6. We note that BadChallenge may output a bit even when there is no bad challenge at all for a given $\mathbf{a}$. These "false positives" are not a problem since, in order to soundly instantiate Fiat-Shamir, we only need the somewhere CI hash function to avoid the bad challenge when it exists.

PARALLEL REPETITIONS. To achieve negligible soundness error, the protocol is repeated $\kappa = \Theta(\lambda)$ times in parallel by first computing $(\mathbf{a}_1, \ldots, \mathbf{a}_\kappa)$ before obtaining the challenge $\mathsf{Chall} = \mathsf{Chall}[1] \ldots \mathsf{Chall}[\kappa]$ and computing the response $\bar{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_\kappa)$, $(z_{\mu,1}, \ldots, z_{\mu,\kappa})$. We then handle $\bar{\mathbf{z}}$ as an integer vector in $\mathbb{Z}^{\kappa\cdot(m+1)}$ and reject it with probability $\theta = \min\left(1, D_{\mathbb{Z}^{2\kappa\cdot(m+1)},\sigma_{\mathrm{eq}}}(\mathbf{z}) / M \cdot D_{\mathbb{Z}^{2\kappa\cdot(m+1)},\sigma_{\mathrm{eq}},\mathsf{Chall}\cdot(\mathbf{1}^\kappa\otimes\mathbf{w})}(\mathbf{z})\right)$, with the value of $M = e^{12/\log(2\kappa\cdot(m+1))+1/(2\log^2(2\kappa\cdot(m+1)))}$. Then, we need to slightly increase σ_{eq} and set $\sigma_{\mathrm{eq}} \geq \log(2\kappa(m+1)) \cdot \sqrt{\kappa(B_r^2 + B_\chi^2)}$.

Lemma 6.6. *The above construction is a trapdoor Σ-protocol for $\mathscr{L}^{\mathrm{eq}}$ if we set $\sigma_{\mathrm{eq}} \geq \log(2m+2) \cdot \sqrt{B_r^2 + B_\chi^2}$ and*

$$B^* > \max(2\sigma_{\mathrm{eq}}\sqrt{2m+2} \cdot (\alpha q\sqrt{m}+1), B_r \alpha q\sqrt{m} + B_\chi).$$

Proof. The special ZK simulator proceeds as follows. Given a statement $(\mathbf{c}_0, \mathbf{c}_1) \in \mathscr{L}^{\mathrm{eq}}_{\mathsf{zk}}$ and a challenge $\mathsf{Chall}^* \in \{0,1\}$, the simulator first samples $\mathbf{z}^* \leftarrow D_{\mathbb{Z}^{2(m+1)},\sigma_{\mathrm{eq}}}$ and $z^*_\mu \leftarrow U(\mathbb{Z}_p)$. Then, it computes

$$\mathbf{a}^* = \mathbf{A}_{\mathrm{eq}} \cdot \mathbf{z}^* + z^*_\mu \cdot [\mathbf{0}^{n\top} \mid p \mid \mathbf{0}^{n\top} \mid p]^\top - \mathsf{Chall} \cdot \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \end{bmatrix} \bmod q.$$

It outputs $(\mathbf{a}^*, \mathsf{Chall}^*, (\mathbf{z}^*, z^*_\mu))$ with probability $1/M$. By construction, the triple $(\mathbf{a}^*, \mathsf{Chall}^*, (\mathbf{z}^*, z^*_\mu))$ satisfies the verification conditions with high probability. We show that it is statistically indistinguishable from a real transcript. If we have $(\mathbf{c}_0, \mathbf{c}_1) \in \mathscr{L}^{\mathrm{eq}}_{\mathsf{zk}}$, there exist $\mathbf{r}_0, \mathbf{r}_1 \in \mathbb{Z}^m$, $\chi_0, \chi_1 \in \mathbb{Z}$ and $\mu \in \mathbb{Z}_p$ such that $\|\mathbf{r}_b\| \leq B_r$, $|\chi_b| \leq B_\chi$ and $\mathbf{c}_b = \bar{\mathbf{A}}_b \cdot [\mathbf{r}_b^\top \mid \chi_b]^\top + \mu \cdot \left[\mathbf{0}^{n\top} \mid p\right]^\top \bmod q$ for $b \in \{0,1\}$. The distribution of $\mathbf{z} \in \mathbb{Z}^{2(m+1)}$ in a real transcript is thus $D_{\mathbb{Z}^{2(m+1)},\sigma_{\mathrm{eq}},\mathsf{Chall}\cdot\mathbf{w}}$, where $\mathbf{w} = [\mathbf{r}_0^\top \mid \chi_0 \mid \mathbf{r}_1^\top \mid \chi_1]^\top$. By Lemma 2.7 and our choice of $\sigma_{\mathrm{eq}} \geq \log(2m+2) \cdot \sqrt{B_r^2 + B_\chi^2}$, the distribution of the simulated $\mathbf{z}^*$ is within statistical distance $2^{-100}/M$ from that of a real non-aborting transcript. Moreover, the component z_μ in the real protocol is uniformly random over $\mathbb{Z}_p$, so is the respective component z^*_μ in the simulation. Finally, in both the real protocol and the simulation, the statement $(\mathbf{c}_0, \mathbf{c}_1)$, the challenge Chall and the response $(\mathbf{z}, z_\mu)$ uniquely determine $\mathbf{a}$.

Soundness can be shown using the same arguments as well. Let us assume that, for a given $\mathbf{a} \in \mathbb{Z}_q^{2(n+1)}$, there exist two valid responses $(\mathbf{z}_b, z_{\mu,b}) \in \mathbb{Z}^{2m+2} \times \mathbb{Z}_p$ with $\|\mathbf{z}_b\| \leq \sigma_{\mathrm{eq}} \cdot \sqrt{2m+2}$ for each $b \in \{0,1\}$ and such that

$$\mathbf{a} + b \cdot \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \end{bmatrix} = \mathbf{A}_{\mathrm{eq}} \cdot \mathbf{z}_b + z_{\mu,b} \cdot \begin{bmatrix} \mathbf{0}^n \\ p \\ \mathbf{0}^n \\ p \end{bmatrix} \bmod q.$$

Subtracting them yields

$$\left[\begin{array}{c}\mathbf{c}_0\\\hline\mathbf{c}_1\end{array}\right] = \mathbf{A}_{\mathrm{eq}}\cdot(\mathbf{z}_1-\mathbf{z}_0) + (z_{\mu,1}-z_{\mu,0} \bmod p)\cdot\left[\begin{array}{c}\mathbf{0}^n\\\hline p\\\hline\mathbf{0}^n\\\hline p\end{array}\right] \bmod q,$$

where $\|\mathbf{z}_1-\mathbf{z}_0\| \le 2\sigma_{\mathrm{eq}}\cdot\sqrt{2m+2}$. Then, we also have

$$\left[\begin{array}{c|c|c|c}-\mathbf{s}_0^\top & 1 & & \\\hline & & -\mathbf{s}_1^\top & 1\end{array}\right]\cdot\left[\begin{array}{c}\mathbf{c}_0\\\hline\mathbf{c}_1\end{array}\right] = \left[\begin{array}{c|c|c|c}\mathbf{e}_0^\top & 1 & & \\\hline & & \mathbf{e}_1^\top & 1\end{array}\right]\cdot(\mathbf{z}_1-\mathbf{z}_0) + (z_{\mu,1}-z_{\mu,0} \bmod p)\cdot\left[\begin{array}{c}p\\\hline p\end{array}\right],$$

which implies that $(\mathbf{c}_0,\mathbf{c}_1)\in\mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}}$ since $|[\,\mathbf{e}_0^\top \mid 1 \mid \mathbf{0}^n \mid 0]^\top\cdot(\mathbf{z}_1-\mathbf{z}_0)] < 2\sigma_{\mathrm{eq}}\sqrt{2m+2}\cdot(\alpha q\sqrt{m}+1) < B^*$ and $|[\mathbf{0}^n \mid 0 \mid \mathbf{e}_1^\top \mid 1]^\top\cdot(\mathbf{z}_1-\mathbf{z}_0)] < B^*$.

We now show that BadChallenge provides the correct output. First, assuming that $(\mathbf{c}_0,\mathbf{c}_1)\notin\mathscr{L}^{\mathrm{eq}}_{\mathsf{sound}}$, for a given $\mathbf{a}\in\mathbb{Z}_q^{2n+2}$, there cannot exist two distinct pairs $(\mu_0,\mathbf{v}_0),(\mu_1,\mathbf{v}_1)\in[-(p-1)/2,(p-1)/2]\times[-B^*/2,B^*/2]^2$ such that the following equality holds over $\mathbb{Z}$ for each $b\in\{0,1\}$:

$$\left[\begin{array}{c|c|c|c}-\mathbf{s}_0^\top & 1 & & \\\hline & & -\mathbf{s}_1^\top & 1\end{array}\right]\cdot\left(\mathbf{a}+b\cdot\left[\begin{array}{c}\mathbf{c}_0\\\hline\mathbf{c}_1\end{array}\right]\right) \bmod q \;=\; \mathbf{v}_b+\mu_b\cdot\left[\begin{array}{c}p\\\hline p\end{array}\right]. \tag{6.14}$$

Let us first assume that there exists no $(\mu_0,\mathbf{v}_0)\in\mathbb{Z}_p\times[-B^*/2,B^*/2]^2$ satisfying (6.14) for $b=0$. Then, there can be no valid response for $\mathsf{Chall}=0$. Indeed, the verifier would only accept a response $(\mathbf{z}_0,z_{\mu,0})\in\mathbb{Z}^{2m+2}\times\mathbb{Z}_p$ satisfying

$$\mathbf{a} = \mathbf{A}_{\mathrm{eq}}\cdot\mathbf{z}_0 + z_{\mu,0}\cdot\left[\begin{array}{c}\mathbf{0}^n\\\hline p\\\hline\mathbf{0}^n\\\hline p\end{array}\right] \bmod q$$

and $\|\mathbf{z}_0\|\le\sigma_{\mathrm{eq}}\sqrt{2m+2}$, which would imply

$$\left[\begin{array}{c|c|c|c}-\mathbf{s}_0^\top & 1 & & \\\hline & & -\mathbf{s}_1^\top & 1\end{array}\right]\cdot\mathbf{a} \bmod q \;=\; \left[\begin{array}{c|c|c|c}\mathbf{e}_0^\top & 1 & & \\\hline & & \mathbf{e}_1^\top & 1\end{array}\right]\cdot\mathbf{z}_0 + z_{\mu,0}\cdot\left[\begin{array}{c}p\\\hline p\end{array}\right]$$

with the inequalities $|[\mathbf{e}_0^\top \mid 1 \mid \mathbf{0}^n \mid 0]\cdot\mathbf{z}_0| < \sigma_{\mathrm{eq}}\sqrt{2m+2}\cdot(\alpha q\sqrt{m}+1) < B^*/2$ and $|[\mathbf{0}^n \mid 0 \mid \mathbf{e}_1^\top \mid 1]\cdot\mathbf{z}_0| < \sigma_{\mathrm{eq}}\sqrt{2m+2}\cdot(\alpha q\sqrt{m}+1) < B^*/2$.

Similarly, assuming that there exists no pair $(\mu_1,\mathbf{v}_1)\in\mathbb{Z}_p\times[-B^*/2,B^*/2]^2$ satisfying (6.14) for $b=1$, we obtain that no valid response can exist for $\mathsf{Chall}=1$. Since there exists $b\in\{0,1\}$ such that no pair $(\mu_b,\mathbf{v}_b)\in[-(p-1)/2,(p-1)/2]\times[-B^*/2,B^*/2]^2$ satisfies (6.14), we conclude that BadChallenge always eliminates a $\mathsf{Chall}\in\{0,1\}$ for which no valid response exists for a given $\mathbf{a}$. □

Chapter

7

Conclusion

Summary of the Contribution

In this thesis, we focused our attention on building new cryptographic primitives, with modern functionalities and advanced security under standard computational assumptions, like learning With Errors (LWE), Decisional Diffie-Hellman (DDH) or Paillier's Decisional Composite Residuosity (DCR) assumption.

In Chapter 3, we discussed the construction of the first Distributed Pseudo-Random Function family that is simultaneously non-interactive and secure under adaptive corruptions. Security is proven under the LWE assumption. The main ingredient for our result is a new security proof for the Key-Homomorphic PRF of [BLMR13], that allows the reduction to know the secret key of the centralized version of the PRF. This feature is important when proving the security in the decentralized case, since a reduction that knows the secrets of all the users, is able to consistently answer adaptive queries for the adversary.

Then we continued with the study of Functional Encryption for the restricted class of linear functions. In Chapter 4 we gave the first construction of MCFE that supports labels, and is secure against adaptive corruptions, under a well established computational assumption. The security is proven under the subexponential hardness of LWE and it exploits, in a non-generic way, the connection with the previously mentioned security proof for DPRFs, against adaptive corruptions.

In Chapter 5, we showed that the IPFE schemes (or some variants) of [ALS16] are in fact adaptive-simulation secure, therefore establishing optimal security for this primitive, among the IND-based and SIM-based definitions considered in the literature so far. The main insight for these proofs was that the semi-adaptive simulator of [Wee17] can be adapted to prove AD-SIM security.

Finally, in Chapter 6, we presented alternative ways to [FLS99, SCO+01] for obtaining unbounded simulation-sound NIZKs, along with a trapdoor Σ-protocol

for proving that two ACPS [ACPS09] ciphertexts encrypt the same message. Combining these two results, we can apply the Naor-Young [NY90] transformation, to get the most efficient public-key encryption scheme with Key-Dependent Message (KDM) security against CCA2 attacks, under a standard assumption (LWE). By using the particular trapdoor Σ-protocol for ACPS ciphertexts and out generic NIZK transformation, we avoid the use of a Karp reduction to the graph Hamiltonicity problem and applying the general transformations of [FLS99, SCO+01].

Open Problems

Below, I propose a list of unsolved questions that naturally arise in the context of this thesis.

Question 1: *Can we get non-interactive DPRFs, secure against adaptive corruptions, under the LWE assumption with a polynomial modulus q ? In general, can we get efficient DPRFs, secure against adaptive corruptions, under any other standard assumption?*

A recent result [Kim20] shows that we can have Key-Homomorphic PRFs from the LWE assumption with polynomial modulus. This implies a generic DPRF construction that only achieves static security under LWE with small modulus.

The construction from [NPR00] is very efficient, but it is only heuristically secure in the static security model, under the assumptions of random oracles.

Question 2: *Can we have AD-SIM functional encryption for quadratic functions, under standard assumptions?*

Practical functional encryption schemes under standard assumptions, that work for functionalities beyond linear functions have been constructed as well [Lin17, BCFG17], but only in the IND model. Recently, [Gay20] gave a semi-AD-SIM secure construction, with succinct ciphertexts, for quadratic functions.

Question 3: *Can we construct MCFE (with labels), for quadratic functions, under standard assumptions?*

Question 4: *Can we push beyond quadratic functionality by constructing FE for degree-*3 *polynomials, under standard assumptions?*

This is one major open problem in functional encryption, as the existence of FE supporting degree-3 polynomials, with succinct ciphertexts, together with the existence of a special class of PRGs, implies [LT17] Indistinguishability Obfuscation (iO) [BGI+01]. This is a very powerful primitive that is known to imply a very large set of cryptographic applications. Moreover,

from a practical point of view we want, efficient schemes that can handle a broad class of functionalitites, from minimal assumptions. For instance, FE for quadratic functions has been used to apply Machine Learning on encrypted data [RDG+19].

Question 5: *Can we build more efficient Correlation-Intractable hash functions, under standard assumptions?*

The Fiat-Shamir transform is usually used to obtain very efficient constructions in the Random Oracle Model. Building very efficient CI hash functions will allow practical schemes, but under standard assumptions. The current standard model CI constructions [PS19] rely on techniques introduced for computing on encrypted data, which are not very efficient. In particular, in the construction of [PS19], the size of the hashing key grows with the description size of the circuits for which it is correlation intractable. It would be desirable to have more compact hashing keys under the standard LWE assumption.

Bibliography

[ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *Eurocrypt*, 2010. Citations: § 13 and 27

[ABDCP15] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC*, 2015. Citations: § 9, 14, and 72

[ABG19] M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner product functional encryption. Cryptology ePrint Archive Report, May 2019. Citations: § 17 and 85

[ABKW19] M. Abdalla, F. Benhamouda, M. Kolhweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC*, 2019. Citations: § 17 and 86

[ABV+12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, 2012. Citations: § 43 and 49

[ACF+18] M Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *Crypto*, 2018. Citations: § 17 and 86

[ACPS09] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Crypto*, volume 5677 of *LNCS*, pages 595–618. Springer, 2009. Citations: § 20, 116, 143, and 149

[AGRW17] M Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *Eurocrypt*, 2017. Citations: § 17, 93, and 96

[AGVW13] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013. Citations: § 14, 15, and 93

[AJW12] G. Asharov, A. Jain, and D. Worichs. Multiparty computation with low communication, computation and interaction via threshold FHE.

Cryptology ePrint Archive: Report 2011/613, 2012. Citations: § 117 and 118

[AKPW13] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *Crypto*, 2013. Citations: § 24

[ALMT20] S. Agrawal, B. Libert, M. Maitra, and R. Titiu. Adaptive simulation security for inner product functional encryption. In *PKC*, 2020. Citations: § 15, 93, 102, and 108

[ALS16] S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products from standard assumptions. In *Crypto*, 2016. Citations: § 15, 17, 57, 74, 93, 94, 96, 97, 102, 103, 108, and 148

[BB04] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Crypto*, 2004. Citations: § 26

[BBL17] F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC*, 2017. Citations: § 81 and 97

[BCFG17] C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functionswith applications to predicate encryption. In *CRYPTO*, 2017. Citations: § 149

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001. Citations: § 13

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC*, 1988. Citations: § 9 and 18

[BGG+18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Crypto*, 2018. Citations: § 12, 59, and 61

[BGGK17] D. Boneh, R. Gennaro, S. Goldfeder, and S. Kim. A lattice-based universal thresholdizer for cryptographic systems. Cryptology ePrint Archive: Report 2017/251, September 2017. Citations: § 59, 60, and 61

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001. Citations: § 149

[BGV11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Cryptology ePrint Archive, Report 2011/277*, 2011. Citations: § 8

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, 2008. Citations: § 9 and 143

[BHY09] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Eurocrypt*, 2009. Citations: § 19, 122, 124, and 125

[BL88] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *Crypto*, 1988. Citations: § 30 and 46

[BLMR13] D. Boneh, K. Lewi, H. Montgomery, and A. Raghunathan. Key-homomorphic PRFs and their applications. In *Crypto*, 2013. Citations: § 11, 12, 13, 35, 36, 37, 38, 40, 41, 42, 43, 44, 46, 49, 65, 74, and 148

[BMR10] D. Boneh, H. Montogomery, and A. Raghunathan. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *ACM-CCS*, 2010. Citations: § 11

[BP14] A. Banerjee and C. Peikert. New and improved key-homomorphic pseudo-random functions. In *Crypto*, 2014. Citations: § 11, 12, 13, 35, 38, 42, 44, and 46

[BPR12] A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In *Eurocrypt*, 2012. Citations: § 11, 13, 35, 42, and 65

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM-CCS*, 1993. Citations: § 12 and 18

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *SAC 2002: 9th Annual International Workshop on Selected Areas inCryptography*, pages 62–75, 2002. Citations: § 9, 115, and 143

[BSW11a] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011. Citations: § 8, 13, 14, 15, 93, 94, 95, 123, and 125

[BSW11b] E. Boyle, G. Segev, and D. Wichs. Fully leakage-resilient signatures. In *Eurocrypt*, 2011. Citations: § 19

[BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proc. of FOCS*, pages 97–106. IEEE Computer Society Press, 2011. Citations: § 8

[CCH+19] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. Rothblum, R. Rothblum, and D. Wichs. Fiat-Shamir: From practice to theory. In *STOC*, 2019. Citations: § 120 and 121

[CDD+99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multi-party computations secure against an adaptive adversary. In *Eurocrypt*, 1999. Citations: § 38

[CDG+18a] J. Chotard, E. S. Dufour, R. Gay, D.-H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *Asiacrypt*, 2018. Citations: § 9, 16, 66, 72, 73, and 74

[CDG+18b] J. Chotard, E. S. Dufour, R. Gay, D.-H. Phan, and D. Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive: Report 2018/1021, 2018. Citations: § 16

[CDS94] R. Cramer, I. Damgård, and B. Schoenmaekers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Crypto*, 1994. Citations: § 117

[CF02] R. Cramer and S. Fehr. Optimal black-box secret sharing over arbitrary abelian groups. In *Crypto*, 2002. Citations: § 30 and 46

[CG99] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen-ciphertext attacks. In *Eurocrypt*, 1999. Citations: § 12

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. Cryptology ePrint Archive, Report 1998/011, 1998. https://eprint.iacr.org/1998/011. Citations: § 12

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. Journal of Cryptology, 25(4), 2010. Citations: § 130

[Cho19] J. Chotard. *Delegation in functional encryption*. PhD thesis, 2019. Citations: § 72 and 73

[CKS00] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in constantinople: practical asynchronous byzantine agreement using cryptography. In *PODC*, 2000. Citations: § 12

[CLW19] R. Canetti, A. Lombardi, and D. Wichs. Fiat-Shamir: From Practice to Theory, Part II (NIZK and Correlation Intractability from Circular-Secure FHE). Cryptology ePrint Archive: Report 2018/1248, 2019. Citations: § 18, 115, 116, 117, 118, 121, 130, 135, and 143

[CM04] M. Chase and S. Meiklejohn. Déjà Q: using dual systems to revisit q-type assumptions. In *Eurocrypt*, 2004. Citations: § 11

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA Conference*, pages 8–26, 2001. Citations: § 13

[Cra96] R. Cramer. Modular design of secure, yet practical cryptographic protocols. PhD thesis, University of Amsterdam, 1996. Citations: § 117

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Crypto*, 1989. Citations: § 12

[DH76] W. Diffie and M. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume 22, pages 644–654, 1976. Citations: § 7 and 8

[DKN+20] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In *CRYPTO*, 2020. Citations: § 44

[Dod00] Y. Dodis. *Exposure-resilient cryptography*. PhD thesis, MIT, 2000. Citations: § 28

[Dod03] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *PKC*, 2003. Citations: § 12

[DT06] I. Damgård and R. Thorbek. Linear integer secret sharing and distributed exponentiation. In *PKC*, 2006. Citations: § 29, 30, 31, 46, 47, and 58

[DY05] Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC*, 2005. Citations: § 11

[DYY06] Y. Dodis, A. Yampolskiy, and M. Yung. Threshold and proactive pseudo-random permutations. In *TCC*, 2006. Citations: § 12

[FHPS13] E. Freire, D. Hofheinz, K. Paterson, and C. Striecks. Programmable hash functions in the multilinear setting. In *Crypto*, 2013. Citations: § 26

[FLS99] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge under general assumptions. *SIAM J. of Computing*, 29(1), 1999. Citations: § 19, 115, 116, 120, 130, 136, 143, 148, and 149

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, 1986. Citations: § 18

[Gay20] Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In *PKC*, 2020. Citations: § 149

[Gen04] R. Gennaro. Multi-trapdoor commitments and their applications to non-malleable protocols. In *Crypto*, 2004. Citations: § 137

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. of STOC*, pages 169–178. ACM, 2009. Citations: § 8 and 143

[GGG+14] S. Goldwasser, S. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *Eurocrypt*, 2014. Citations: § 15 and 16

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. Citations: § 14

[GGHZ14] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Fully secure functional encryption without obfuscation. Cryptology ePrint Archive: Report 2014/666, 2014. Citations: § 14

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *J. of ACM*, volume 33, 1986. Citations: § 7, 8, and 11

[GKL+14] S. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive: Report 2013/774, 2014. Citations: § 66

[GKP+13] S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proc. of STOC*, pages 555–564. ACM Press, 2013. Citations: § 14

[GKPV10] S. Goldwasser, Y. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the Learning with Errors assumption. In *ICS*, 2010. Citations: § 24 and 44

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229, 1987. Citations: § 8

[GMY03] J. Garay, P. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In *Eurocrypt*, 2003. Citations: § 137

[GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM-CCS*, 2006. Citations: § 13

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. of STOC*, pages 197–206. ACM, 2008. Citations: § 13, 22, 123, 126, 130, and 136

[GSW13] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Crypto*, number 8042 in LNCS, pages 75–92, 2013. Citations: § 8 and 44

[GTK03] S. Goldwasser and Y. Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, 2003. Citations: § 18

[GVW12] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Crypto*, 2012. Citations: § 14

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC*, 2013. Citations: § 13

[GVW15a] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *Crypto*, 2015. Citations: § 13

[GVW15b] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, 2015. Citations: § 59, 60, and 61

[HHP06] S. Hoory, A. Hager, and T. Pitassi. Monotone circuits for the majority function. In *APPROX-RANDOM*, 2006. Citations: § 31, 48, and 49

[HILL99] J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 8(4):1364–1396, 1999. Citations: § 11

[HLL16] S. Han, S. Liu, and L. Lyu. Efficient kdm-cca secure public-key encryption for polynomial functions. In *ASIACRYPT* , 2016. Citations: § 116

[Jag15] T. Jager. Verifiable random functions from weaker assumptions. In *TCC*, 2015. Citations: § 26 and 27

[Kil06] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC*, 2006. Citations: § 123

[Kim20] Sam Kim. Key-homomorphic pseudorandom functions from LWE with a small modulus. In *Advances in Cryptology – EUROCRYPT 2020*, pages 576–607, 2020. Citations: § 12 and 149

[KMT19] Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. Simple and efficient kdm-cca securepublic key encryption. In *ASIACRYPT*, 2019. Citations: § 116

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Eurocrypt*, 2008. Citations: § 13

[KT18] F. Kitagawa and K. Tanaka. A framework for achieving kdm-cca secure public-keyencryption. In *ASIACRYPT*, 2018. Citations: § 116

[KY16] S. Katsumata and S. Yamada. Partitioning via non-linear polynomial functions: More compact IBEs from ideal lattices and bilinear maps. In *Asiacrypt*, 2016. Citations: § 27

[Lin17] H. Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In *CRYPTO*, 2017. Citations: § 149

[LNPT19] B. Libert, K. Nguyen, A. Passelègue, and R. Titiu. Simulation-sound arguments for LWE and applications to KDM-CCA2 security, 2019. https://eprint.iacr.org/2019/908. Citations: § 19, 115, 142, and 143

[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*, 2010. Citations: § 13

[LST18] B. Libert, D. Stehlé, and R. Titiu. Adaptively secure distributed PRFs from LWE. In *TCC*, 2018. Citations: § 13, 35, and 46

[LT17] H. Lin and S. Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *CRYPTO*, 2017. Citations: § 149

[LT19] B. Libert and R. Titiu. Multi-client functional encryption for linear functions in the standard model from lwe. 11923:520–551, 2019. Citations: § 17, 65, 72, and 85

[LW09] A. Lewko and B. Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *ACM-CCS*, 2009. Citations: § 11

[Lyu12] V. Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Eurocrypt*, 2012. Citations: § 23

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Proc. of EUROCRYPT*, pages 700–718. Springer, 2012. Citations: § 23, 24, 44, 74, 77, and 129

[MR07] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput*, 37(1):267–302, 2007. Citations: § 22, 23, 77, and 81

[MS95] S. Micali and R. Sidney. A simple method for generating and sharing pseudo-random functions. In *Crypto*, 1995. Citations: § 12

[MY04] P. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In *Eurocrypt*, 2004. Citations: § 137

[Nie02] J.-B. Nielsen. A threshold pseudorandom function construction and its applications. In *Crypto*, 2002. Citations: § 12

[NPR00] M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Eurocrypt*, 2000. Citations: § 11, 12, 41, 72, and 149

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, 1997. Citations: § 11 and 12

[NRR00] M. Naor, O. Reingold, and A. Rosen. Pseudo-random functions and factoring. In *STOC*, 2000. Citations: § 11

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, 1990. Citations: § 18, 116, 142, 143, and 149

[O'N10] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/. Citations: § 8, 13, 14, 15, and 93

[Ore87] Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs. *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 462–471, 1987. Citations: § 18

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of EUROCRYPT*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999. Citations: § 15 and 32

[Pol00] J. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13:433–447, 2000. Citations: § 98

[PS19] C. Peikert and S. Shiehian. Non-interactive zero knowledge for NP from (plain) Learning With Errors. In *Crypto*, 2019. Citations: § 18, 19, 115, 116, 121, 136, 143, and 150

[RDG+19] Theo Ryffel, Edouard Dufour-Sans, Romain Gay, Francis Bach, and David Pointcheval. Partiallyencrypted machine learning using functional encryption. In *Advances in Neural InformationProcessing Systems (NeurIPS 2019),*, 2019. Citations: § 150

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005. Citations: § 31, 77, 126, and 143

[RSA78] Ronald L. Rivest, A. Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. In *Communications of the Association for Computing Machinery*, volume 21, pages 120–126, 1978. Citations: § 8

[Sah99] A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999. Citations: § 18, 120, and 142

[Sch87] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53(2-3):201–224, 1987. Citations: § 77

[SCO+01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero-knowledge. In *Crypto*, 2001. Citations: § 19, 20, 115, 116, 120, 143, 148, and 149

[Sha48] E. Claude Shannon. A mathematical theory of communication. In *Bell system technical journal*, volume 27, pages 379–423, 1948. Citations: § 9

[Sha79] A. Shamir. How to share a secret. In *Communications of the ACM*, 1979. Citations: § 28

[SW05] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Eurocrypt*, 2005. Citations: § 13

[Tho09] R. Thorbek. *Linear Integer Secret Sharing*. PhD thesis, Department of Computer Science - University of Arhus, 2009. Citations: § 29 and 30

[Val84] L. Valiant. Short monotone formulae for the majority function. *J. of Algorithms*, 3(5), 1984. Citations: § 31

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt*, 2005. Citations: § 44

[Wat15] B. Waters. A punctured programming approach to adaptively secure functional encryption. In *Proc. of CRYPTO*, LNCS. Springer, 2015. Citations: § 14

[Wee14] H. Wee. Dual system encryption via predicate encoding. In *TCC*, 2014. Citations: § 81 and 97

[Wee17] H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In *TCC*, 2017. Citations: § 15, 93, 96, 97, and 148

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th FOCS*, pages 162–167, 1986. Citations: § 8