UNIVERSITÉ DU
LUXEMBOURG

PhD-FSTC-2017-08
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defence held on 23/01/2017 in Luxembourg

to obtain the degree of

DOCTEUR DE L’UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Massimo Chenal
Born on 15 May 1987 in Aosta (Italy)

Key-Recovery Attacks Against Somewhat Homomorphic Encryption Schemes

Dissertation defence committee

Dr. Peter Y.A. Ryan, dissertation supervisor
Professor, University of Luxembourg

Dr. Qiang Tang
Research Associate, Luxembourg Institute of Science and Technology

Dr. Jean-Sébastien Coron, chairman
Associate Professor, University of Luxembourg

Dr. Jintai Ding
Professor, University of Cincinnati

Dr. Johannes A. Buchmann, deputy chairman
Professor, Technische Universität Darmstadt

Acknowledgements

I wish to thank many people who shared with me their knowledge, expertise and time, provided advice, encouragement and friendship.

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Dr. Peter Y. A. Ryan, for his kindness and advice throughout all the course of my PhD. These four years of working under his guidance were a truly invaluable and rewarding experience. I am grateful to my co-supervisor, Dr. Qiang Tang, for his support of my PhD studies and related research, for suggesting me the topic of this dissertation and for steering me in the right direction. His guidance helped me throughout all my research and the writing process of this thesis, and the discussions that we had were always very beneficial and provided me with new insights and ideas. I wish to thank both of you for letting me part of the APSIA group, and for your support and mentoring which made this dissertation possible. I would also like to thank Prof. Dr. Johannes Buchmann for his insightful comments and encouragement, which incentivized me to widen my research from various perspectives.

I wish to thank my fellow doctoral students and colleagues from APSIA for their friendship, for the stimulating and helpful discussions, and for all the nice moments we have had during these four years. Thanks to all my friends in Luxembourg for making these four years really special.

My PhD studies were supported by Luxembourg National Research Fund (FNR), under the AFR Grant No. 6996691, and in several occasions by the Doctoral School of Computer Science and Computer Engineering (DS-CSCE) of the University of Luxembourg.

Abstract

In 1978, Rivest, Adleman and Dertouzos introduced the concept of privacy homomorphism and asked whether it is possible to perform arbitrary operations on encrypted ciphertexts. Thirty years later, Gentry gave a positive answer in his seminal paper at STOC 2009, by proposing an ingenious approach to construct fully homomorphic encryption (FHE) schemes. With this approach, one starts with a somewhat homomorphic encryption (SHE) scheme that can perform only limited number of operations on ciphertexts (i.e. it can evaluate only low-degree polynomials). Then, through the so-called bootstrapping step, it is possible to turn this SHE scheme into an FHE scheme. After Gentry's work, many SHE and FHE schemes have been proposed; in total, they can be divided into four categories, according to the hardness assumptions underlying each SHE (and hence, FHE) scheme: hard problems on lattices, the approximate common divisor problem, the (ring) learning with errors problem, and the NTRU encryption scheme. Even though SHE schemes are less powerful than FHE schemes, they can already be used in many useful real-world applications, such as medical and financial applications. It is therefore of primary concern to understand what level of security these SHE schemes provide. By default, all the SHE schemes developed so far offer IND-CPA security - i.e. resistant against a chosen-plaintext attack - but nothing is

said about their IND-CCA1 security - i.e. secure against an adversary who is able to perform a non-adaptive chosen-ciphertext attack. Considering such an adversary is in fact a more realistic scenario.

Gentry emphasized it as a future work to investigate SHE schemes with IND-CCA1 security, and the task to make some clarity about it was initiated by Loftus, May, Smart and Vercauteren: at SAC 2011 they showed how one family of SHE schemes is not IND-CCA1 secure, opening the doors to an interesting investigation on the IND-CCA1 security of the existing schemes in the other three families of schemes. In this work we therefore continue this line of research and show that most existing somewhat homomorphic encryption schemes are not IND-CCA1 secure. In fact, we show that these schemes suffer from key recovery attacks (stronger than a typical IND-CCA1 attack), which allow an adversary to recover the private keys through a number of decryption oracle queries. The schemes, that we study in detail and for which we develop key recovery attacks, are the ones in the three categories mentioned above. Our key recovery attacks work in such a way that a malicious attacker can recover the private key of an underlying encryption scheme completely, bit by bit, when he's allowed a given number of decryption oracle accesses. As a result, this dissertation shows that all known SHE schemes fail to provide IND-CCA1 security.

While it is true that IND-CPA security may be enough to construct cryptographic protocols in presence of semi-honest attackers, key recovery attacks will pose serious threats for practical usage of SHE and FHE schemes: if a malicious attacker (or a compromised honest party) submits manipulated ciphertexts and observes the behavior (side channel leakage) of the decryptor, then it may be able to recover all plaintexts in the system. Therefore, it is very desirable to design SHE and FHE with IND-CCA1 security, or at least design them to prevent key recovery attacks. This raises the interesting question whether it is possible or not to develop such IND-CCA1 secure SHE scheme. Up to date, the only positive result in this direction is a SHE scheme proposed by Loftus et al. at SAC 2011 (in fact, a modification of an existing SHE scheme and IND-CCA1 insecure). However, this IND-CCA1 secure SHE scheme makes use of a non standard knowledge assumption, while it would be more interesting to only rely on standard assumptions. We propose then a variant of the SHE scheme proposed by Lopez-Alt, Tromer, and Vaikuntanathan at STOC 2012, which offers good indicators about its possible IND-CCA1 security.

In the conclusion, we finish our dissertation with some interesting future directions which could expand from our current work.

Table of contents

Nomenclature

Acronyms

AGCD	Approximate Greatest Common Divisor problem
BDDP	Bounded Distance Decoding Problem
CCA1	Non-Adaptive Chosen-Ciphertext Attack
CCA2	Adaptive Chosen-Ciphertext Attack
CPA	Chosen-Plaintext Attack
CVP	Closest Vector Problem
FHE	Fully Homomorphic Encryption
GCD	Greatest Common Divisor
GLWE	General Learning With Errors problem
HNF	Hermite Normal Form
IND-CCA1	Indistinguishability under CCA1
IND-CPA	Indistinguishability under CPA
KDM	Key Dependent Message

LLL	Lenstra-Lenstra-Lovasz algorithm
LWE	Learning With Errors problem
NIZK	Non-Interactive Zero Knowledge proof
NTRU	The NTRU Cryptosystem
PACD	Partial Approximate Greatest Common Divisor problem
PA	Plaintext-Awareness
PCP	Polynomial Coset Problem
PIR	Private Information Retrieval
PLWE	Polynomial Learning with Errors problem
RLWE	Ring Learning With Errors Problem
RSA	The RSA Cryptosystem
SHE	Somewhat Homomorphic Encryption
SPIP	Small Principal Ideal Problem
SSSP	Sparse Subset-Sum Problem
SVP	Shortest Vector Problem

Homomorphic encyptions schemes

BGV12	The encryption scheme from [12]
BLLN13	The encryption scheme from [9]
Bra12	The encryption scheme from [11]

BV11a	The encryption scheme from [14]
BV11b	The encryption scheme from [15]
DGHV10	The encryption scheme from [75]
GSW13	The encryption scheme from [44]
LTV12	The encryption scheme from [55]

Chapter 1

Introduction

In public-key cryptography, a sender wants to transmit a message through a public channel to a receiver, and they have to be sure that the communication is kept hidden from eavesdroppers. The idea was first introduced by Diffie and Hellman in [29], and the first instantiation was obtained by Rivest, Shamir and Adleman in the breakthrough paper [70] where the sender encrypts a message with the receiver's public key, and the latter decrypts with his secret key.

In 1978, Rivest, Adleman and Dertouzos [69] wondered whether it is possible to perform operations on encrypted ciphertexts, by means of what they defined to be a *privacy homomorphism*. Homomorphic encryption is therefore a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. Rivest, Adleman and Dertouzos asked: ”can we do *arbitrary* computations on data while it remains encrypted, without ever decrypting it?”. This way, data can be kept confidential while being processed, which enables useful tasks to be accomplished with data residing in environments which are not trusted. This is a hugely valuable capability in a world of heterogeneous networking and distributed computation. Although it was immediately recognized as a very interesting possibility in cryptography, for the next thirty years no concrete construction was built, and finding a general method for computing on encrypted data had been a major goal in cryptography studies ever since. A number of homomorphic encryption schemes has been developed during the years, but they supported essentially only one basic operation (addition or multiplication).

Several homomorphic encryption schemes have been developed during the years, but their homomorphic capabilites were limited to just a single operation: this is the case, for instance, of the ElGamal cryptosystem [32]. Other famous examples include the basic RSA cryptosystem [70], which is homomorphic with respect to multiplication, and the Paillier cryptosystem [66], which is homomorphic with respect to addition. In all these cases, the encryption schemes are homomorphic with respect to a single algebraic operation. While it is true that these simple homomorphic cryptosystems have a wide range of applications (for example, private information retrieval or secure voting), being restricted to a single operation renders them incapable of evaluating more general transformations on encrypted data. Therefore, the natural question that Rivest, Adleman, and Dertouzous raised in [69] just after the discovery of RSA in 1978, was whether there existed an encryption scheme that was *fully* homomorphic, namely, homomorphic with respect to both addition and multiplication.

It was only in 2009 that the quest for a first viable fully homomorphic encryption (FHE) scheme scheme ended, when Craig Gentry settled this conjecture [37]. It turned out then that privacy homomorphisms encryption schemes are actually possible to achieve; using modern terminology, we call them *fully homomorphic encryption*, or FHE, schemes. He used ideal lattices to propose an ingenious approach to construct FHE schemes. These are encryption systems that permit arbitrarily complex computations on encrypted data. The development of FHE is a revolutionary advance: it greatly extends the scope of the computations which can be applied to process encrypted data homomorphically.

A natural application that we can think of FHE (and SHE, as we will define later) is outsourcing computation to an untrusted third party, such as a cloud server. Such a scenario arises when a client does not have the computational resources to carry out a computation on his own, and therefore needs to outsource or delegate the computation to a third party, which is potentially untrustworthy. In an outsourcing computation example using FHE, the client first encrypts his input using a given FHE scheme; then he sends the ciphertexts to the cloud, which will perform the computation homomorphically. Finally, the client receives the response which is encrypted, but that he can decrypt to learn the result of the computation. See Figure 1.1.

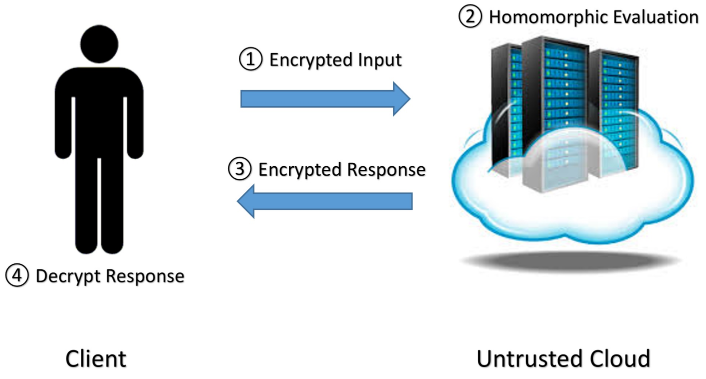


Fig. 1.1 An application of FHE - Outsourcing computations to Cloud

An Example: Alice's Jewelry Store. To understand intuitively how FHE works, let's start from an example as explained by Gentry in [36].

Alice is the owner of a jewelry store, and she gives precious raw materials (diamonds, gold, silver, etc.) to her workers in order to be assembled into rings and necklaces. However she does not trust her workers, assuming that they could steal her jewels if given the chance. Her goal therefore is to let the workers *process* the raw material into the desired finished pieces, but without giving them *access* to these materials.

That's how she solves this problems, and at the same time this solution gives to us the intuition behind FHE schemes. She orders a transparent and impenetrable glovebox, which is secured by a lock for which only she has the key. After putting the precious raw materials inside the glovebox, she locks it and let the workers access to the closed glovebox. Using the gloves, the workers can manipulate the pieces and craft the desired necklaces and bracelets inside the box. Now the workers can do this without having the possibility to directly access, or steal, the precious raw materials since they are locked inside the impenetrable glovebox. After the work is done, the workers return the glovebox to Alice, who can then open the box thanks to her key and get the desired necklace. To sum up, the workers process the raw materials into the desired piece of art, without having direct access to these materials.

For the analogy, the non penetrable glovebox, with the raw materials locked inside, represent an encryption of some initial data $m_1, \ldots, m_n$, which can be accessed only with Alice's given secret decryption key sk. The gloves represent the homomorphism, or *malleability*, of the encryption scheme; they allow the ”raw” data to be manipulated while it is inside the ”encryption box”. The final piece of art (necklace or bracelet) inside the box represents the encryption $\mathsf{Enc}(f(m_1, \ldots, m_n))$, where f is the desired function of the initial data. In this case, the lack of access is represented by the lack of physical access to the jewels.

Of course this analogy does not faithfully represent some aspects of homomorphic encryption, but as long as it is not taken too literally, it gives a good rough idea of how FHE works.

Since Gentry published his idea there has been huge interest in the area, with regard to improving the schemes, implementing them and applying them. Several other FHE schemes have been released, improving asymptotic efficiency; nevertheless, existing FHE all follow the same blueprint as the one in Gentry's original scheme. In short, ciphertexts produced by an FHE scheme can be operated on in such a way that we obtain a ciphertext that corresponds to the addition or multiplication of the respective plaintexts. The ability to algebraically operate over ciphertexts is of great importance because any algorithm can be transformed into a sequence of additions and multiplications in $\mathbb{Z}_2$. Therefore, such a scheme can evaluate any algorithm solely with access to the encryption of its input, and such that the computation returns the encryption of the output. Since Gentry's work, many FHE constructions have appeared in the literature. Following Gentry blueprint for building a FHE scheme, all the subsequent constructions start with a *somewhat homomorphic encryption* (SHE) scheme that can perform only limited number of operations on ciphertexts (i.e. it can evaluate only low-degree polynomials). Then, through the so-called bootstrapping step, we can turn this SHE scheme into an FHE scheme. So, a SHE scheme allows a fixed number of multiplications of ciphertexts; these schemes are building blocks for the FHE schemes and provide much better efficiency guarantees than their FHE counterparts. Therefore, they are used already in a number of practical applications, as we will see later on in this chapter. Note that researchers have proposed the concept of leveled FHE schemes (e.g. [12, 44]), which allow third parties to evaluate any circuits up to a certain depth. Often in the following discussion, we treat these schemes as SHE.

Back to Alice's Jewelry Store. In her store, Alice notices that after a worker uses the gloves for one minute, the gloves stiffen and become unusable. This defective glovebox can be seen as a SHE scheme. After a given number of operations on ciphertexts have been done, no more operations can be performedor the resulting ciphertext would fail to decrypt to the correct plaintext. As we will see later, the way SHE and FHE schemes work is by adding noise to ciphertexts. By performing operations on them, this noise grows. Homomorphic multiplication increases the noise much more than addition; a SHE scheme can evaluate only low-degree polynomials over encrypted data, i.e. it can perform only a limited number of additions and multiplications. Eventually, the noise makes the resulting ciphertext so noisy that it is not possible to decrypt correctly anymore.

Back to Alice, she really would like to avoid this situation, because some of her more complicated pieces like necklaces and bracelets may take up to an hour to be assembled, while the gloves stiffen only after one minute of usage. She needs to find out a way to use these defective boxes to get the workers to securely assemble even the most complicated pieces. She notices that the defective boxes have anyway a interesting property that might be useful: they have a one-way insertion slot, like the ones we can see in the post office mail bins. But they are also flexible enough so that it is possible to put one box inside another through the slot. This property plays a key role in the solution of this problem, which leads us to the concept of *bootstrappable* homomorphic encryption scheme.

Here is the idea: she gives a worker a glovebox, which we call box #1, containing all the raw materials. She also gives him several extra gloveboxes: box #2 which contains (locked inside) the key opening box #1; box #3 which contains the key opening box #2; and so on. To assemble an intricate piece, the worker manipulates the materials in box #1 until the gloves stiffen. Then, he puts box #1 inside box #2, where the latter box already contains the key opening box #1. Using the gloves for box #2, he opens box #1 with this key, extracts the (partially) assembled piece, and then he continues to assemble pieces within box #2 until its gloves stiffen. In a similar fashion, he then places box #2 inside box #3, and so on he continues in this way. When the worker finally finishes his work inside box #n, he gives the box to Alice. Of course, this system is succesfull only if the worker can open box #i within box #$(i + 1)$, after which he still has the time to make an extra progress on the assembly, before the gloves of box #$(i + 1)$ stiffen. So that's why it is important that the unlocking operation (and the extra assembly work) takes less than a minute. If she has enough defective gloveboxes, then it is possible to assemble any piece, no matter how complicated they are.

In this analogy, the defective gloveboxes represent our SHE scheme, which can perform additions and multiplications on ciphertexts for a while until the noise becomes too large and prevents the scheme to continue making operations on ciphertexts. What we would like to do is use this SHE scheme to construct a FHE scheme. As before, box #1 with the precious raw materials inside represents the ciphertexts that encrypt the initial data. Box #$(i + 1)$ with the key for box i inside represents an encrypted secret decryption key. Alice understands that there is only one thing that her workers need to be able to do in less than one minute with the gloves, together with performing a small operation on the piece: unlock box $\#i$ within box $\#(i+1)$ and extract the piece. It will turn out that there is only one function that our scheme (call it $\mathcal{E}$) needs to be able to handle, with a some room left over to perform one more addition or multiplication: the decryption function (which is like unlocking the ”encryption box”). If $\mathcal{E}$ has this self-referential property of being able to handle its own decryption function (augmented by a single gate), we say that it is *bootstrappable*. As it turns out, if $\mathcal{E}$ is bootstrappable, then one can use $\mathcal{E}$ to construct a FHE scheme. We will see more on this in Section 3.2.1.

However, all the proposals have a common drawback: they are not practical. Initially, the algorithms involved in the constructions, although having polynomial complexity, had high polynomial degree. Later, the asymptotic complexity became much better. Indeed, we now have constructions with polylogarithmic overhead per operation, but with terribly high constants. Although FHE is not practical yet, many constructions have been proposed recently, achieving a SHE scheme. They allow a limited depth of operations to be performed. These constructions are indeed very useful in practice, specially in order to provide security in the scenario of cloud computing. A SHE scheme is important also in the implementation of private information retrieval (PIR) protocols, which can be seen as a building block to the solution for the privacy problem that emerges when we give our data to the cloud. We will discuss in a moment more in detail the practical applications of FHE and SHE schemes.

In the cloud computing scenario it is natural to imagine an attacker having access to a decryption oracle (e.g., the cloud can feed invalid ciphertexts to a user and monitor their behaviour). It is obvious that a homomorphic encryption scheme cannot have security of ciphertexts under adaptive attacks. Hence, adaptive attacks are already a very serious concern in this setting. But one could hope that at least the private key remains secure in the presence of a decryption oracle.

Main Result and Contribution

With this work, we show that most - if not all - the SHE schemes built so far are not secure even against non-adaptive chosen ciphertext attacks.

In Chapter 4 we will see how all the SHE and FHE schemes can be divided into four main families, according to the hardness assumptions they rely on. This categorization is made clear with figure 4.1. As we will see better in Chapter 5, our Thesis is inspired by the work of [54], whose authors were the first to observe adaptive *key recovery attacks* by showing that one of the above mentioned families is vulnerable to such attacks. Thanks to such a technique, one can completely determine the secret key of a given SHE scheme in a CCA1 scenario attack. In this dissertation we continue this line of work by presenting original and efficient key recovery attacks for most of the existing SHE schemes (see figure 4.1). This shows that IND-CCA1 security is hard to achieve in homomorphic encryption.

It is important to have a clear idea of what level of security is offered by the known SHE schemes; in fact, adaptive key recovery attacks on homomorphic encryption seem to be realistic in certain scenarios, so they are potentially a serious problem in practice. The only homomorphic encryption scheme known to resist such attacks is the a variation of a known scheme presented by Loftus et al. [54]; however, the authors make use of a non-standard lattice assumption, which makes this SHE scheme not a very good candidate for a IND-CCA1 secure scheme.

We are going to see in the next chapter a few practical applications of FHE and SHE.

Chapter 2

Practical Applications of SHE and FHE

Even though SHE schemes are less powerful than FHE schemes, they find already many useful interesting real-world applications, ranging from medical to financial applications [61]. Currently homomorphic encryption is rather slow, but it can be used already for several practical uses. Concrete practical applications and concrete useful functions can be computed, and most of them only require a limited number of multiplications of ciphertexts (and a normally very large number of additions of ciphertexts). For these applications, it is often enough to consider an implementation of a SHE scheme.

2.1 Applications that are Feasible Today

Among the applications that we can consider by using the technology that we have nowadays, medical applications are the most interesting. As it is well explained in [61], consider a cloud service which manages electronic medical records, and a scenario where the medical data of a patient is continuously uploaded to a service provider in an encrypted form. The cloud service constantly collect important health information (like for example blood pressure, heart rate, weight and blood sugar). In this case, the user is the data owner, and therefore the data is encrypted under his public key and only him - the user - can decrypt. The service provider computes on the encrypted data, runs some statistics thanks to which it can predict the likelihood of certain medical conditions that can occur, or simply just keep track of the user's health. Obviously, the main benefit for the user's perspective is to allow real-time health analysis based on readings from various sources without having to disclose this data to any one source. Also we have to keep in mind that, since the volume of the data involved is generally large, it is better for the user not to store and manage all this data locally on a given device, but instead to use cloud storage and computation.

A scenario like the above one requires computing simple statistical functions (i.e. mean, standard deviation, logistical regressions, etc.) that are typically used for prediction of likelihoods of certain outcomes. For these functions, it is enough to only consider a SHE scheme which computes many additions and a small number of multiplications on ciphertexts: for example, averages require no multiplications, standard deviation requires only one multiplication, and logistical regression requires a few multiplications. For a concrete instantiation of such an application, we can mention the actual implementation [50] by Lauter at Eurocrypt 2015, where she described of a heart attack prediction by Microsoft.

Among other interesting applications, one can consider consumer privacy in advertising, data mining, financial privacy, and forensic image recognition. See [61] and [4] for further details.

2.2 Constructions that use FHE and SHE Schemes as Building Blocks

Homomorphic encryption schemes can be used to construct cryptographic tools such as the following.

- **Zero Knowledge Proofs**: As shown by Gentry in [36], we can use homomorphic encryption in order to build non-interactive zero knowledge (NIZK) proofs of small size. Suppose that a user would like to prove knowledge of a satisfying assignment of bits $b_1, \ldots, b_t$ for a given boolean circuit $\mathcal{C}$. The NIZK proof is as follow: generate a public key, encrypt the b_i's, and homomorphically evaluate $\mathcal{C}$ on these encryptions. A standard NIZK proof is attached to prove that every ciphertext encrypts either 0 or 1, and that the output of the evaluation encrypts 1.

- **Delegation of Computation**: Besides outsourcing data, another important aspect of cloud computing is given by outsourcing computation. Consider the scenario in which a user want to delegate the computation of a function f to a server. The user, however, does not exclude the possibility that the server may be malicious, or just not working properly. In other words, the user may not trust the result of the server's computation. The user demands to have a proof that the computation was done correctly and verifying this proof should also be significantly more efficient than the user doing the computation. Chung et al. [21] used FHE to design schemes for delegating computation. One example for the delegation of computation is message authenticators. Consider a user who outsources computation on a data set and wants to check that the return value is really the correct result. The tag should be independent of the size of the original data set, and only verifiable for the holder of the private key. Gennaro and Wichs propose such a scheme in [35] based on a FHE scheme. However, it only supports a bounded number of verification queries.

- **Multiparty Computation**: Multiparty computation requires interaction between participants. Damgard et al. [28] provide a description of how a SHE scheme can be used to construct offline multiplication during the computations. The players use the SHE scheme in a preprocessing phase, but return to the much more efficient techniques of multiparty computation in the computation phase.

- **Signatures**: Gorbunov et al. [46] presented a construction of levelled fully homomorphic signature schemes. The scheme can evaluate arbitrary circuits with maximal depth d over signed data and homomorphically produce a short signature which can be verified by anybody using the public verification key. The user uploads the signed data x, then the server runs some function g over the data which yields $y = g(x)$. Additionally, the server publishes the signature $\sigma_{g,y}$ to verify the computation. This work also introduces the notion of homomorphic trapdoor functions (HTDF), one of the building blocks for the signature construction. HTDF themselves are based on the small integer solution (SIS) problem.

2.2.1 Implementations of FHE schemes

Some of the FHE schemes have already been implemented. Among them we find:

- HElib: it is a software library implementing the [12] scheme, with optimizations to run homomorphic evaluation faster (i.e., the Smart-Vercauteren ciphertext packing techniques [72], and the Gentry-Halevi-Smart optimizations [43]). See https://github.com/shaih/HElib.
- ”Stanford FHE”: it is a working implementation of the scale-invariant leveled homomorphic encryption system in [11]. See https://crypto.stanford.edu/people/dwu4/fhe-project.html.
- Implementation of [75] FHE scheme over the integers. Implementation is describer in [24]. See https://github.com/coron/fhe.
- Implementation SEAL of the scheme in [9]. The implementation by Microsoft Research is described in [62].

2.3 Impact of Key Recovery Attacks

For all these reasons, it is important to have a clear idea of what security level is offered by the known SHE schemes, and in particular to understand whether a given SHE scheme is secure against key recovery attacks. As we mentioned, in theory IND-CPA security may be enough for us to construct cryptographic protocols, in particular if we assume semi-honest attackers. However, key recovery attacks will pose serious threat for practical usage of SHE schemes if an attacker becomes malicious (or, an honest party is compromised) and submits manipulated ciphertexts to observe the behavior of the decryptor. We illustrate this point by presenting an ”attack” against the LWE-based single-server private information retrieval (PIR) protocol in [15].

The PIR protocol is very simple: the client has a long-term key tuple for a SHE scheme and a secret key sk for a symmetric encryption scheme; a PIR query is an encrypted index under sk; a PIR response is a ciphertext under the SHE public key, generated by the server (who is given the ciphertext of sk under the SHE public key) by homomorphically evaluating the encrypted index and the database; the client obtains the desired bit by decrypting the ciphertext using the SHE private key.

Clearly, if the server is malicious, then it can mount a key recovery attack by manipulating the responses and monitoring the client’s behavior. With the SHE private key, the server can recover all the private information of the client. In order to prevent the attack, the client can require the server to prove all computations are done properly. However, this might make the server’s computational complexity very heavy and make the protocol less efficient than others.

As we just saw, we can consider cloud among the most straightforward applications. As more and more data is outsourced into cloud storage, often unencrypted, considerable trust is required in the cloud storage providers.[1]

[1] The Cloud Security Alliance lists data breach as the top threat to cloud security [47]. Encrypting the data with conventional encryption avoids the problem. However, now the user cannot operate on the data and must download the data to perform the computations locally. With FHE the cloud can perform computations on behalf of the user and return only the encrypted result.

2.4 Structure of the Thesis

In Chapter 3 we introduce the notation and the standard definitions that we will use throughout our dissertation; in particular we will cover the definitions of homomorphic encryption (both fully and somewhat), as well as security definitions. Then, in Chapter 4 we present the literature review focused on the hardness assumptions underlying the known SHE and FHE schemes; we will also describe the original construction of the lattice-based FHE scheme given by Gentry in [37], as well as some optimization and modifications. Then we will continue on Chapter 5 by describing the main results of our contributions, i.e. key-recovery attacks against all the schemes in categories (2), (3) and (4) (see Figure 4.1). As we will see, all the SHE schemes developed so far have been shown to be vulnerable against key-recovery attacks, with the exception of one scheme proposed in [54]. This SHE, however, makes use of a non standard knowledge assumption.

More in particular, in Chapter 5 we continue the line of work of [54, 76] to present key recovery attacks for the schemes [15, 14, 44, 11]. Our attacks can also be applied to the SHE scheme in [12]. We also develop a new key recovery attack against the SHE scheme in [75], and our attack is more efficient and conceptually simpler than that from [76]. Our results essentially show that the SHE schemes underlying the FHE schemes in category (2) and (3) in the figure 4.1 are not IND-CCA1 secure. Previous analysis had not paid much attention to the NTRU-based SHE schemes (to be defined in Section 4.1.6), i.e. the schemes in category (4) in 4.1. Two representative schemes in this line are those by Lopez-Alt, Tromer and Vaikuntanathan [55] and Bos et al. [9]. We have proposed efficient key recovery attacks against these two NTRU-based SHE schemes. This shows that the SHE schemes underlying the FHE schemes in category (4) in the figure 4.1 are not IND-CCA1 secure.

It would be interesting to obtain one IND-CCA1 secure SHE scheme, and in Chapter 6 we show how our variant of the [55] scheme offer good indications that it is IND-CCA1 secure. More in particular, we will consider the SHE scheme from [55] - for which we develop an efficient key recovery attack in Chapter 5 - and we tweak its decryption step in two ways, leading to scenarios (2) and (3) as explained in that chapter. We successfully show a key-recovery attack for scenario (2); however, scenario (3) seems to resist any attempt to show a key-recovery attack. We will show that our usual strategy for key-recovery attacks does not lead to a successful attack, and our variation of the [55] SHE scheme seems therefore to be the a good candidate for being IND-CCA1 secure.

In Chapter 7 we conclude our work and suggest interesting future works and directions.

Chapter 3

Notation and Standard Definitions

3.1 Preliminaries

Let $\mathbb{N}$ be the set of natural numbers, $\mathbb{Z}$ the ring of integers, $\mathbb{Q}$ the field of rational numbers, and $\mathbb{F}_q$ a finite field with q elements, where q is a power of a prime p. In particular, we will consider often $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z} = \mathbb{Z}_p$. If $r \in \mathbb{Z}_q$, we indicate as r^{-1} its inverse in $\mathbb{Z}_q$, i.e. that value such that $r^{-1} \cdot r = 1 \bmod q$. For integers $a, q \in \mathbb{Z}$ sometimes we use notation $[a]_q := a \bmod q$. For a given integer n and a ring R, we indicate $\mathcal{M}_n(R)$ as the ring of square $n \times n$ matrices with entries in R. For a ring R and a (two-sided) ideal I of R, we consider the quotient ring R/I. For given vectors $\mathbf{v} := (v_1, \ldots, v_n), \mathbf{w} := (w_1, \ldots, w_n) \in \mathbb{R}^n$, we let $||\mathbf{v}|| := \sqrt{v_1^2 + \cdots + v_n^2}$ be the Euclidean norm of its coefficients, $||\mathbf{v}||_\infty := \max\{|v_1|, \ldots, |v_n|\}$ be its maximum norm, and we indicate with $< \mathbf{v}, \mathbf{w} >:= \sum_i v_i w_i$ the dot product of $\mathbf{v}, \mathbf{w}$. For a given rational number $x \in \mathbb{Q}$, we let $\lfloor x \rceil$, $\lfloor x \rfloor$ and $\lceil x \rceil$ be respectively the rounding function, the floor function and the ceiling function. For a given integer $n \in \mathbb{N}$, $\lfloor n + 1/2 \rceil = n + 1$. To indicate that an element a is chosen uniformly at random from a set A we use notation

$a \overset{\$}{\leftarrow} A$. For a set A, we let its cardinality be $|A|$. We consider also the standard basis $\{\mathbf{e}_i\}_{i=1}^n$ of $\mathbb{R}^n$, where the coefficients of $\mathbf{e}_i$ are all 0 except for the i-th coefficient, which is 1. We denote the map that reduces an integer x modulo q and uniquely represents the result by an element in the interval $(-q/2, q/2]$ by $[\cdot]_q$. Therefore, we will consider the ring $\mathbb{Z}_q$ as $\mathbb{Z}_q := \{-\left\lfloor \frac{q}{2} \right\rfloor, -\left\lfloor \frac{q}{2} \right\rfloor + 1, \dots, \left\lfloor \frac{q}{2} \right\rfloor\}$. We extend this map to polynomials in $\mathbb{Z}[X]$ and thus also to elements of R by applying it to their coefficients separately; given a polynomial $a(x) \in R$, we define the map

$$[\cdot]_q : R \to R, \quad a(x) = \sum_{i=0}^{n-1} a_i x^i \mapsto \sum_{i=0}^{n-1} [a_i]_q x^i$$

Let f, g be two functions defined on some subset of $\mathbb{R}$. We write $f(x) \in O(g(x))$ (resp. $f(x) \in \Omega(g(x))$) if, informally, $|f(x)| \leq g(x) \cdot k$ (resp. $|f(x)| \geq g(x) \cdot k$), for some positive k. We write $f(x) = \widetilde{O}(g(x))$ if $f(x) = O(g(x) \log^k g(x))$ for some k. We write $f(x) \in \omega(g(x))$ if $|f(x)| \geq k \cdot |g(x)|$, for every fixed positive number k. Finally, we have $f(x) \in \Theta(g(x))$ if $g(x) \cdot k_1 \leq f(x) \leq g(x) \cdot k_2$ for some positive k_1, k_2.

We also recall some terminology from circuit theory. The depth of a circuit is the maximum distance from an input gate to an output gate; the i-th level of a circuit consists of all gates with depth i; the size of a circuit is the number of gates it contains.

Unless otherwise specified, λ will always denote the security parameter of the encryption schemes. In the asymmetric schemes we are going to discuss, the secret key will be denoted as sk, and the public key will be pk.

3.2 Homomorphic Encryption

The following definitions are adapted from [37]. We only assume bit-by-bit public-key encryption, i.e. we only consider encryption schemes that are homomorphic with respect to boolean circuits consisting of gates for addition and multiplication mod 2. Extensions to bigger plaintext spaces and symmetric-key setting are straightforward; we skip the details.

Definition 1 (Homomorphic Encryption)**.** A public key homomorphic encryption (HE) scheme is a set $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Encrypt}_{\mathcal{E}}, \mathsf{Decrypt}_{\mathcal{E}}, \mathsf{Evaluate}_{\mathcal{E}})$ of four algorithms, all of which must run in polynomial time. When the context is clear, we will often omit the index $\mathcal{E}$.

$\mathsf{KeyGen}(\lambda) = (\mathsf{sk}, \mathsf{pk})$

- input: λ
- output: sk; pk

$\mathsf{Encrypt}(\mathsf{pk}, m) = c$

- input: pk and plaintext $m \in \mathbb{F}_2$
- output: ciphertext c

$\mathsf{Decrypt}(\mathsf{sk}, c) = m'$

- input: sk and ciphertext c
- output: $m' \in \mathbb{F}_2$

$\mathsf{Evaluate}(\mathsf{pk}, C, (c_1, \dots, c_r)) = c_e$

- input: pk, a circuit C, ciphertexts $c_1, \dots, c_r$, with $c_i = \mathsf{Encrypt}(\mathsf{pk}, m_i)$
- output: ciphertext c_e

All these steps must be efficient - that is, the computational complexity of all of these algorithms must be polynomial in security parameter λ and the size of C.

Definition 2 (Correct Homomorphic Decryption)**.** The public key homomorphic encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ is correct for a given t-input circuit C if, for any key-pair $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{KeyGen}(\lambda)$, any t plaintext bits $m_1, \ldots, m_t$, and any ciphertexts $\overline{c} = (c_1, \ldots, c_t)$ with $c_i \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}, m_i)$, $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Evaluate}(\mathsf{pk}, C, \overline{c})) = C(m_1, \ldots, m_t)$ holds.

Definition 3. The scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ is homomorphic for a class $\mathcal{C}$ of circuits if it is correct for all circuits $C \in \mathcal{C}$. We say that $\mathcal{E}$ is a fully homomorphic encryption (FHE) scheme if it is correct for all boolean circuits.

Informally, a homomorphic encryption scheme that can perform only a limited number of operations is called a somewhat homomorphic encryption (SHE) scheme. We will make this point more clear now. The way known FHE schemes work is by adding some noise ciphertexts and, by performing operations on them, this noise grows. Therefore it is often convenient to work with homomorphic encryption schemes that can perform only a limited number of operations; as we have seen in Chapter 1, these are informally called somewhat homomorphic encyption (SHE) schemes. We remark that homomorphic multiplication increases the noise significantly more than addition; SHE schemes can evaluate only low-degree polynomials over encrypted data, i.e. they can perform only a limited number of additions and multiplications. Eventually, the noise makes the resulting ciphertext so noisy that it is not possible to decrypt correctly anymore.

A family of schemes $\{\mathcal{E}^{(L)} : L \in \mathbb{Z}^+\}$ is *leveled fully homomorphic* if they all use the same decryption circuit, $\mathcal{E}^{(L)}$ is homomorphic for all circuits of depth at most L (that use some specified set of gates), and the computational complexity of $\mathcal{E}^{(L)}$'s algorithms is polynomial (the same polynomial for all L) in λ, L and (in the case of $\mathsf{Evaluate}_{\mathcal{E}(L)}$) the size of the circuit C.

As defined above, FHE can be obtained from any secure encryption scheme by an algorithm $\mathsf{Evaluate}$ that attaches a description of the circuit C to the ciphertext tuple, and a $\mathsf{Decrypt}$ procedure that first decrypts all the ciphertexts and then evaluates C on the corresponding plaintext bits. Two properties of homomorphic encryption, circuit-privacy and compactness, avoid this solution.

Definition 4. The scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ is *compact* if there exists a fixed polynomial bound $b(\lambda)$ so that for any key-pair $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{KeyGen}(\lambda)$, any circuit C and any sequence of ciphertext $\overline{c} = (c_1, \ldots, c_t)$ that was generated with respect to pk, the size of the ciphertext $\mathsf{Evaluate}(\mathsf{pk}, C, \overline{c})$ is not more than $b(\lambda)$ bits (independently of the size of C).

Moreover, we want that a FHE scheme is efficient also in the evaluation step, i.e. the complexity of $\mathsf{Evaluate}$ must depend only polynomially on the security parameter. Therefore, two extra requirements must be satisfied in order to have a FHE scheme:

- Decrypting c_e (the ciphertext output by $\mathsf{Evaluate}_{\mathcal{E}}$) must take the same amount of computation as decrypting c (a ciphertext output by $\mathsf{Encrypt}_{\mathcal{E}}$). Moreover, we require that c_e has the same size as c (compact ciphertext requirement).

  The size of c_e, as well as the time needed to decrypt it are completely independent of f, the evaluation function. Also the complexities of $\mathsf{Decrypt}_{\mathcal{E}}, \mathsf{KeyGen}_{\mathcal{E}}$ and $\mathsf{Encrypt}_{\mathcal{E}}$, are polynomial in λ.

- The $\mathsf{Evaluate}_{\mathcal{E}}$ step must be efficient, as follows (see [38]). Let S_f be the size of a boolean circuit that computes f. Then $\mathsf{Evaluate}_{\mathcal{E}}$ is said to be efficient if there is a polynomial g such that, for any function f that is represented by a circuit of size S_f, $\mathsf{Evaluate}_{\mathcal{E}}(\mathrm{pk}, f, c_1 \dots, c_t)$ has complexity at most $S_f \cdot g(\lambda)$.

3.2.1 From SHE to FHE

All FHE schemes today follow the same blueprint of the scheme developed by Gentry in 2009 [37]. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ be an homomorphic encryption scheme. For a given λ, a secret key sk and two ciphertexts c_1, c_2, the set of augmented decryption circuits consists of the following two circuits:

$$\begin{aligned} C_{\text{add}}(\mathsf{sk}, c_1, c_2) &:= \mathsf{Decrypt}(\mathsf{sk}, c_1) + \mathsf{Decrypt}(\mathsf{sk}, c_2) \bmod 2 \\ C_{mult}(\mathsf{sk}, c_1, c_2) &:= \mathsf{Decrypt}(\mathsf{sk}, c_1) \cdot \mathsf{Decrypt}(\mathsf{sk}, c_2) \bmod 2 \end{aligned}$$

We denote this set by $D_{\mathcal{E}}(\lambda) := (C_{\text{add}}, C_{\text{mult}})$. Now, for every value of the security parameter λ let $C_{\mathcal{E}}(\lambda)$ be a set of circuits with respect to which $\mathcal{E}$ is correct. We say that $\mathcal{E}$ is *bootstrappable* if $D_{\mathcal{E}}(\lambda) \subseteq C_{\mathcal{E}}(\lambda)$ holds for every λ. In other words, a SHE scheme is bootstrappable if it can evaluate $D_{\mathcal{E}}(\lambda)$; that is, it is able to evaluate its own decryption function plus an additional operation. This means the decryption function of the SHE scheme can be expressed as polynomial of degree low enough to be handled within the homomorphic capacity of the SHE scheme, with enough capacity left over to evaluate a NAND gate.

Gentry's blueprint comes in three points.

1. We build a SHE scheme.

2. If the SHE is already bootstrappable, go to point (3). Otherwise, we make it bootstrappable by *squashing* the decryption circuit of the SHE scheme: we transform the scheme into one with the same homomorphic capacity but a decryption circuit that is simple enough to allow bootstrapping. Gentry showed a way to do this by adding a 'hint' about the secret key to the evaluation key: a sparse set of values, a (secret) subset of which sums up to the secret key.

3. To obtain a FHE, there exists a bootstrapping theorem which states that given a bootstrappable SHE scheme, one can transform it into a leveled FHE scheme. Moreover, if the bootstrappable scheme is semantically secure, then also the leveled FHE is. See [37] for further details.

   Furthermore, if the scheme satisfies *circular security* (which allow to safely encrypt the leveled FHE secret key under its own public key) - we obtain a pure (and not just leveled) FHE scheme. Bootstrapping "refreshes" a ciphertext by running the decryption function on it homomorphically, using an encrypted secret key (given in the evaluation key; it is the hint generated in the squashing step). This re-encryption process produces a new, more compact and less noisy encryption of the original plaintext; this way, the ciphertext can be evaluated in more additions and multiplications without making the noise grow too much.

We remark that the squashing step introduces a new hardness assumption, and therefore one might want to avoid this. It turns out that the squashing step is not necessary, and indeed in [15, 39] the authors are able to avoid it. It is also possible to obtain a leveled FHE without the bootstrapping step, see [12].

3.3 Security Definitions

We now define the main security notions used in the cryptographic schemes. There exist several security definitions, but for FHE schemes normally one is concerned mainly with IND-CPA and IND-CCA1 security. We do not focus on IND-CCA2 security since it cannot be satisfied by FHE schemes. We cover also key-recovery attacks.

The security of a public-key encryption scheme in terms of indistinguishability is normally presented as a game between a challenger and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The scheme is considered secure if no adversary can win the game with significantly greater probability than an adversary who must guess randomly. The game runs in two stages:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
- $(m_0, m_1) \leftarrow \mathcal{A}_1^{(\cdot)}(\mathsf{pk})$ /* Stage 1 */
- $b \leftarrow \{0, 1\}$
- $c^* \leftarrow \mathsf{Encrypt}(m_b, \mathsf{pk})$
- $b' \leftarrow \mathcal{A}_2^{(\cdot)}(c^*)$ /* Stage 2 */

The adversary is said to win the game if $b = b'$, with the advantage of the adversary winning the game being defined by

$$\mathrm{Adv}_{\mathcal{A},\mathcal{E},\lambda}^{\text{IND-atk}} = |\Pr(b = b') - 1/2|$$

A scheme is said to be IND-atk secure if no polynomial time adversary $\mathcal{A}$ can win the above game with non-negligible advantage in the security parameter λ. The precise security notion one obtains depends on the oracle access one gives the adversary in its different stages:

- If $\mathcal{A}$ has access to no oracles in either stage then atk=CPA (indistinguishability under chosen plaintext attack)
- If $\mathcal{A}$ has access to a decryption oracle in stage one then atk=CCA1 (indistinguishability under non-adaptive chosen ciphertext attack)
- If $\mathcal{A}$ has access to a decryption oracle in both stages then atk=CCA2, often now denoted simply CCA (indistinguishability under adaptive chosen ciphertext attack)
- If $\mathcal{A}$ has access to a ciphertext validity oracle in both stages, which on input of a ciphertext determines whether it would output $\perp$ or not on decryption, then atk=CVA.

We have

$$\text{IND-CCA2} \Rightarrow \text{IND-CCA1} \Rightarrow \text{IND-CPA}$$

According to the definition, in order to show that a scheme is not IND-CCA1 secure, we only need to show that an adversary can guess the bit b with a non-negligible advantage given access to the decryption oracle in Stage 1. Formally, in a *key recovery attack*, an adversary can output the private key given access to the decryption oracle in Stage 1. In comparison, a key recovery attack is stronger than a typical IND-CCA1 attack, and can result in more serious vulnerabilities in practice.

A key recovery attack allows an attacker to recover the private key of an underlying encryption scheme when given a number of decryption oracle accesses. In the literature, all SHE schemes have been developed with the aim of being IND-CPA secure (resistant against a chosen-plaintext attack). In [37], Gentry emphasized it as a future work to investigate SHE schemes with IND-CCA1 security (i.e. secure against a non-adaptive chosen-ciphertext attack). Up to now, the only scheme proven IND-CCA1 secure is that by Loftus et al. [54], but it relies on some non-standard knowledge assumptions (see [54], for details). Most works in this direction focus on devising attacks against existing SHE schemes. Our main contribution is to show that most existing SHE schemes suffer from key recovery attacks, which allow an attacker to recover the private key of an underlying encryption scheme when given a number of decryption oracle accesses. It is clear that a key recovery attack is stronger than a typical attack against IND-CCA1 security. At this moment, we have the following results:

- No malleable cryptosystem (in particular, no SHE and FHE scheme) can be IND-CCA2 secure. The reason is straightforward, based on the fact that the adversary is allowed to manipulate the challenged ciphertext, submit it to the decryption oracle in an IND-CCA2 attack, and recover the plaintext.

- It is still unknown if there exists a FHE scheme which satisfy IND-CCA1 security. However, we can already say that, if such a FHE scheme exists, it cannot follow Gentry's blueprint: any FHE scheme that adopts Gentry's bootstrapping technique cannot be CCA-1 secure, because the bootstrapping technique requires one to publish the encryption of the secret keys: the private key is encrypted and the adversary is able to submit the ciphertext to the decryption oracle.

- Loftus et al. [54] showed that Gentry's SHE scheme [37] is not IND-CCA1 secure and presented an IND-CCA1 attack against the variation proposed in [40]. They also showed that the same attack applies to the other variant by Smart and Vercauteren [71]. In fact, the attacks are both key recovery attacks. Moreover, they modified the SHE in [71] and proved its IND-CCA1 security based on a new assumption.

In theory, IND-CPA security may be enough for us to construct cryptographic protocols, in particular if we assume semi-honest attackers. However, key recovery attacks will pose serious threat for practical usage of SHE and FHE. If a malicious attacker submits manipulated ciphertexts and observes the behavior (side channel leakage) of the decryptor, then it may be able to recover all plaintexts in the system. Therefore, it is very desirable to design SHE and FHE with IND-CCA1 security, or at least to avoid key recovery attacks.

It is interesting to consider also security in the context of *Plaintext Awareness*, introduced by Bellare and Rogaway [6] in the random oracle model, and later refined into the security notions PA-0, PA-1 and PA-2 by Bellare and Palacio [5]. Intuitively a scheme is said to be PA if the only way an adversary is able to create a valid ciphertext is by applying encryption to a public key and a valid message. In [5] it is proven that if a scheme is PA-1 (resp. PA-2) and at the same time IND-CPA, then it is actually secure against IND-CCA1 (resp. IND-CCA2) attacks. Plaintext awareness can be also explained intuitively by considering that if the adversary knows the plaintext underlying each ciphertext it produces, then he has no need for a decryption oracle. Therefore, PA and IND-CPA must imply IND-CCA1.

The advantage of the results of [5] is that we work in the standard model to prove security of a scheme; however, this comes with the price that we need to make a strong assumption to prove a scheme is PA-1 or PA-2. The assumption required is a so-called *knowledge assumption*. For example, in the context of encryption schemes supporting a single homomorphic operation, the authors of [5] show that the Cramer-Shoup Lite scheme [25] and an ElGamal variant introduced by Damgard [27] are both PA-1, and hence IND-CCA1, assuming the standard DDH (to obtain IND-CPA security) and a Diffie-Hellman knowledge assumption (to obtain PA-1 security). (Informally, the Diffie-Hellman knowledge assumption is the assumption that an algorithm can only output a Diffie-Hellman tuple if the algorithm ”knows” the discrete logarithm of one-tuple member with respect to another.) See [5] for details, or e.g. [54] for a high-level explanation. In [54], the authors follow work from [5] in order to show that a variation of the Smart-Vercauteren SHE scheme [71] achieves PA1 security, therefore showing that the scheme achieves PA1 + CPA $\Rightarrow$ CCA1 security. We will come back to this in a later section.

3.3.1 Circuit Privacy

An important requirement for a FHE scheme is that $\mathsf{Encrypt}_{\mathcal{E}}$ and $\mathsf{Evaluate}_{\mathcal{E}}$ have the same output distribution (computationally indistinguishable). This is called *circuit privacy*: roughly, this means that the ciphertext output of $\mathsf{Evaluate}_{\mathcal{E}}$ reveals nothing about the circuit $\mathcal{C}$ that it evaluates beyond the output value of that circuit, even for someone who knows the secret key.

Definition 5. A homomorphic encryption scheme $\mathcal{E}$ is *circuit-private* for given circuits in $\mathcal{C}_{\mathcal{E}}$ if, for any pair of keys $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{KeyGen}_{\mathcal{E}}(\lambda)$, any circuit $C \in \mathcal{C}_{\mathcal{E}}$, and any fixed ciphertexts $c_1, \ldots, c_t$ that are in the image of $\mathsf{Encrypt}_{\mathcal{E}}$ for plaintexts $m_1, \ldots, m_t$ the following distributions are computationally indistinguishable:

$$\mathsf{Encrypt}_{\mathcal{E}}(\mathsf{pk}, C(m_1, \ldots, m_t)) \approx \mathsf{Evaluate}_{\mathcal{E}}(\mathsf{pk}, C, (c_1, \ldots, c_t))$$

3.3.2 Circular Security: A Special Case of KDM Security.

Circular security is required in order to obtain FHE from SHE. It is a special case of the more general notion of *key dependent message* (KDM) security. Informally, an encryption scheme is KDM secure if an adversary cannot distinguish the encryption of a key-dependent message from an encryption of 0. Let $n > 0$ be an integer and let $\mathcal{F}$ be a finite set of functions $\mathcal{F} := \{f : S^n \to M\}$ For each function $f \in \mathcal{F}$ we require that $|f(z)|$ is the same for all inputs $z \in S^n$ (i.e. the output length is independent of the input).

KDM security is defined with respect to $\mathcal{F}$ using the following game that takes place between a challenger and an adversary $\mathcal{A}$ [8]. For an integer $n > 0$ and a security parameter λ the game proceeds as follows.

- The challenger chooses a random bit $b \overset{\$}{\leftarrow} \{0, 1\}$. It generates $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n)$ by running KeyGen n times, and sends the vector $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ to $\mathcal{A}$

- The adversary repeatedly issues queries where each query is of the form (i, f) with $1 \leq i \leq n$ and $f \in \mathcal{C}$. The challenger responds by setting

$$y \leftarrow f(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \in M \quad \text{and} \quad c \overset{\$}{\leftarrow} \begin{cases} \mathsf{Encrypt}(\mathsf{pk}_i, y) & \text{if } b = 0 \\ \mathsf{Encrypt}(\mathsf{pk}_i, 0^{|y|}) & \text{if } b = 1 \end{cases}$$

  and sends c to $\mathcal{A}$

- Finally, the adversary outputs a bit $b' \in \{0, 1\}$

We say that $\mathcal{A}$ is a $\mathcal{F}$-KDM adversary and that $\mathcal{A}$ wins the game if $b = b'$. Define $\mathcal{A}'$s advantage as

$$\mathrm{Adv}^{\mathrm{KDM}^{(n)}}_{\mathcal{A},\mathcal{E},\lambda} := |\Pr(b = b') - 1/2|$$

Chapter 4

General Introduction to Existing SHE and FHE Schemes

As we mentioned, the SHE schemes (and by extension, the FHE schemes) can be categorized into four main families of schemes, according to the hardness assumptions they rely on (see Figure 4.1). In this chapter we are going to investigate in more detail these hardness assumptions. We also describe somehow informally in Section 4.2 the idea behind the original FHE scheme [37], and the variation made by [71, 40].

After Gentry's work, many SHE and FHE schemes have been proposed. Based on the underlying hardness assumptions, these schemes can be categorized and divided in four categories, as follows.[1]

(1) The first category starts with Gentry [36, 37]. A number of variations, optimizations and implementations appear in [71, 40]. The security of these schemes are based on hard problems on lattices, more specifically on ideal lattices. These are the initial schemes, that still

[1] It is worth mentioning that Nuida [65] proposed a new framework for noise-free FHE, based on finite non-commutative groups. This is completely different from everything appeared in literature so far, since the ciphertext in all known schemes carry some noise. Nevertheless, a secure instantiation has yet to be found.

depend on the hard lattice problems, like the Sparse Subset Sum Problem (SSSP).

(2) The second category starts with van Dijk et al. [75]. More variants, implementation and optimizations appear in [23, 24, 20]. The security of these schemes rely on the approximate greatest common divisor (AGCD) problem and some variants. These schemes are based on integers, and they are the simplest to understand.

(3) The third category starts with Brakerski and Vaikuntanathan [15, 14]. More variants appear in [61, 12, 42, 11, 44]. The security of these schemes are based on the learning with errors (LWE) and on the ring-learning with errors (RLWE) problems. These schemes bring new concepts and allow better constructions in practice. We have also asymptotically better constructions that are based on the approximate eigenvector method [44, 16].

(4) The fourth category of schemes is based on the NTRU encryption scheme [48]. In [55] it is shown how to obtain a homomorphic encryption scheme in a multi-user setting, introducing the notion of multi-key homomorphic encryption where it is possible to compute any function on plaintexts encrypted under multiple public keys. The multi-key FHE of [55] is based on the NTRU scheme [48] and on ideas introduced in [12]. NTRU-based schemes permit to obtain ciphertexts that correspond to just one ring element, simplifying previous schemes. NTRU-based SHE offers the possibility of encoding integers in a natural way, that can be used to solve practical problems such as statistical applications [51, 10].

See Fig. 4.1 for a graphical visualization of the main families.

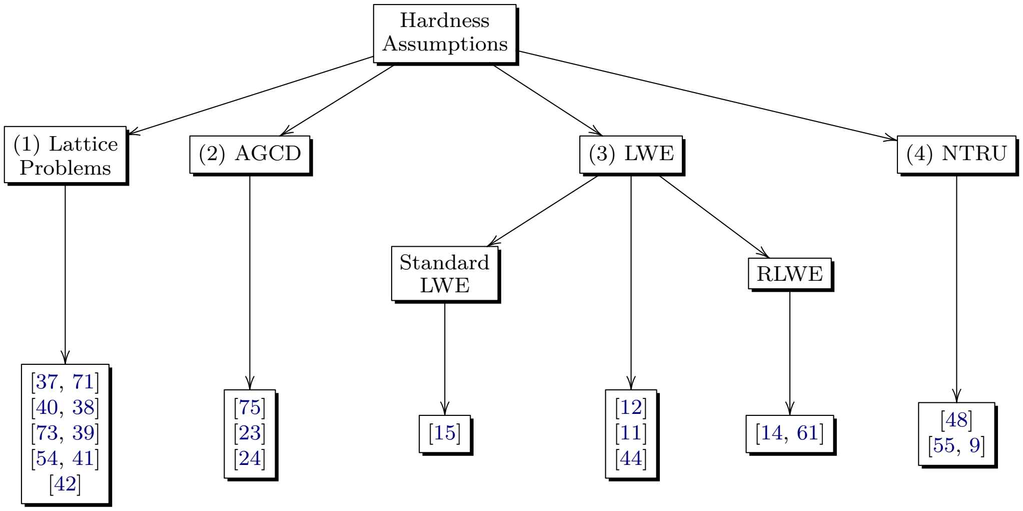


Fig. 4.1 Hardness assumptions and relevant papers

4.1 Hardness Assumptions

Before going any further, we list the hardness assumptions that rely on the homomorphic schemes that we are going to discuss in the next section. (See figure 4.1.)

4.1.1 Approximate Greatest Common Divisor (AGCD)

This hardness assumption is found in the family of schemes started by van Dijk, Gentry, Halevi and Vaikuntanathan (the DGHV10 scheme [75]). For a specific (η-bit) odd positive integer p, and an integer ρ (the size of the noise), consider the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \{\text{choose } q \xleftarrow{\$} \mathbb{Z}\cap[0, 2^{\gamma}/p), r \xleftarrow{\$} \mathbb{Z}\cap(-2^{\rho}, 2^{\rho}) : \text{output } x = pq+r\}$$

Definition 6 ((ρ, η, γ)-approximate GCD)**.** Given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$ for a randomly chosen η-bit odd integer p, output p.

Here the goal is to recover a secret number p (typically a large prime number), given polynomially many near-multiples $x_0, \ldots, x_m$ of p, that is, each integer x_i is of the hidden form $x_i = pq_i + r_i$ where each q_i is a very large integer and each r_i is a very small integer. The hardness of approximate integer common divisors problems, was introduced in 2001 by Howgrave-Graham. Several lattice-based approaches have been developed in order to solve the approximate GCD problem. All attacks are very inefficient if we choose the parameters wisely; namely whenever $\#\mathrm{bits}(q_i) \gg \#\mathrm{bits}(p)^2$.

In [23], the authors use a partial version of this problem (PACD): here, the setting is exactly the same, except that x_0 is chosen as an exact multiple of p, namely $x_0 = pq_0$ where q_0 is a very large integer chosen such that no non-trivial factor of x_0 can be found efficiently: for instance, they select q_0 as a rough number, i.e. without any small prime factor. More precisely, let p be a η-bit prime integer, and let q_0 a be a random square-free 2^λ-rough integer in $[0, 2^\gamma/p)$. Consider

$$\mathcal{D}'_\rho(p, q_0) = \{q \xleftarrow{\$} \mathbb{Z} \cap [0, q_0), r \xleftarrow{\$} \mathbb{Z} \cap (-2^\rho, 2^\rho) : \text{output } x = pq + r\} \subseteq \mathcal{D}_{\gamma,\rho}(p)$$

Definition 7 (Error-free (ρ, η, γ)-approximate GCD)**.** Given $x_0 := q_0 \cdot p$ and polynomially many samples from $\mathcal{D}'_\rho(p, q_0)$, output p.

The version in [23] was used to build a more efficient variant of the DGHV10 FHE scheme. In [17], Y. Chen and P.Q. Nguyen presented a new PACD algorithm whose running time is $2^\rho/2$ polynomial-time operations, which is essentially the square root of that of GCD exhaustive search (which at the time was the best attack available). This directly leads to a new GACD algorithm running in $2^{3\rho/2}$ polynomial-time operations, and allowed them to experimentally break the FHE challenges proposed in [23].[2]

4.1.2 Lattice-Based Assumptions

Lattice-based primitives are interesting because:

- their security can be based on worst-case hardness assumptions;
- they appear to remain secure even against quantum computers;
- they can be quite efficient;
- for FHE no other cryptographic assumption is known to suffice.

Lattice problems are in general easy to solve if the algorithm is provided with a "good" basis (see definition later). Lattice reduction algorithms aim, given a basis for a lattice, to output a new basis consisting of relatively short, nearly orthogonal vectors. The LLL algorithm [52] was an early efficient algorithm for this problem which could output an almost reduced lattice basis in polynomial time. In the late 1990s, several new results on the hardness of lattice problems were obtained; e.g. Ajtai [1] found a

[2] The AGCD problem was first studied by Howgrave-Graham [49]. In that paper, as well as in [75], several possible lattice attacks on this problem were listed, including using orthogonal lattices and the so-called Coppersmith's method. Further cryptanalytic work was done by [17, 22, 24] and, more recently, by J. Ding and C. Tao in [30]: here the authors propose a new algorithm for solving the general approximate common divisors (GACD) problems, which is based on lattice reduction algorithms on certain special lattices and linear equation solving algorithms over integers. They propose an algorithm which can solve the problem with some special parameters in polynomial time.

However, for general parameters the AGCD problem and its variants are still believed to be hard. See [33] for a survey on the topic and a comparison on the known lattice algorithms for the AGCD problem.

surprising worst-case/average-case connection for certain lattice problems. In particular, two lattice-based public-key cryptosystems were published: the Ajtai-Dwork cryptosystem (AD) [2] and the Goldreich-Goldwasser-Halevi cryptosystem (GGH) [45]:

AD: It is provably secure unless a worst-case lattice problem (a variant of the SVP, see later) can be solved in probabilistic polynomial time.

GGH: It relies on the CVP (see later).

There is no proven worst-case/average-case property for the GGH, but it is more practical than AD: given the security parameter λ, key-size and encryption times are $O(\lambda^2)$ and $O(\lambda^4)$ for GGH and AD respectively.

Nguyen showed in [63] that the GGH cryptosystem is insecure: any ciphertext reveals information on the plaintext, and decrypting ciphertexts can be reduced to a particular CVP, easier than the general one. He was able to solve four out of five numerical challenges proposed by the authors of [45]; they also proposed a modified secure scheme, but impractical.

We start by recalling some notions from lattice theory. For more details see, for instance, [64]. A *lattice* of $\mathbb{R}^n$ is a discrete subgroup of $(\mathbb{R}^n, +)$. Let $\mathbf{b}_1, \ldots, \mathbf{b}_m$ be arbitrary vectors in $\mathbb{R}^m$. Denote by $\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ the set of all integral linear combinations of the $\mathbf{b}_i$'s: $\Lambda := \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m) = \{\sum_{i=1}^m n_i \mathbf{b}_i : n_i \in \mathbb{Z}\}$. Λ is a lattice if $\mathbf{b}_i \in \mathbb{Q}^n$, $\forall i$ or $\mathbf{b}_i \in \mathbb{R}^n$ are linearly independent. When $\Lambda = \mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is a lattice, we say that Λ is spanned by the b_i's, and that the b_i's are *generators*. When the b_i's are further linearly independent, we say that $(\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is a *basis* of the lattice Λ. In practice, we will usually restrict to integral lattices, so that the underlying matrices are integral matrices. The *dimension* or *rank* of a lattice Λ in $\mathbb{R}^n$, denoted by $\dim(\Lambda)$, is the dimension d of its linear span denoted by span(Λ). The lattice is said to be *full-rank* when $d = n$: in the following, we will consider only full-rank lattices (dimension n). It can be shown that $|\det(B_i)|$ is constant for all basis B_i of Λ, and we call it the *determinant* $\det(\Lambda)$ of Λ. To a basis B of a lattice Λ it can be associated the half-open parallelepiped

$$\mathcal{P}(B) = \{\sum_{i=1}^{n} x_i \mathbf{b_i} : x_i \in [-1/2, 1/2)\}$$

$\mathcal{P}(B)$ is called the *fundamental parallelepiped* associated to B. Moreover, every element of $\mathbb{R}^n/\Lambda$ has a unique representative in $\mathcal{P}(B)$. We have that vol($\mathcal{P}(B)$) =det(Λ).

In lattice problems it is particular important a specific basis of a lattice, the so called *Hermite normal form* (HNF) basis. Every full-rank lattice has a unique HNF basis: a non-singular square matrix $H = (h_{i,j})$ with integer entries is in HNF if

- H is lower triangular: $h_{i,j} = 0$ for all $i < j$
- its diagonal entries, are positive: $h_{i,i} > 0$ for all i,
- in a given column, the entries below the diagonal are non-negative and less than the diagonal: for $j > i$, $h_{i,i} > h_{j,i} \geq 0$.

Given any basis B of Λ, one can compute HNF(Λ) efficiently via Gaussian elimination. Given a specific basis B of a lattice Λ, in lattice problems often one is asked to reduce a vector $\mathbf{c} \in \mathbb{R}^n$ modulo this basis; we denote it with $\mathbf{c} \bmod B$. It is the unique vector $\mathbf{c}' \in \mathcal{P}(B)$ such that $\mathbf{c} - \mathbf{c}' \in \Lambda$. Given $\mathbf{c}$ and B, $\mathbf{c}' \bmod B$ can be computed efficiently as

$$\mathbf{c} - \lfloor \mathbf{c} \cdot B^{-1} \rceil \cdot B = [\mathbf{c} \cdot B^{-1}] \cdot B$$

In lattice problems, shortest vectors are of central importance. The length of the shortest nonzero vector in a lattice Λ is denoted $\lambda_1(\Lambda)$. Generalizing, one can define the *successive minima*: for any lattice Λ and integer $k \leq \text{rank}(\Lambda)$, let $\lambda_k(\Lambda)$ be the smallest $r > 0$ such that Λ contains at least k linearly independent vectors of length bounded by r.

Ideal Lattices

Gentry's SHE scheme is a scheme over *ideal lattices*. Let $f(x)$ be an integer monic irreducible polynomial of degree n, e.g. $f(x) = x^n + 1$, where n is a power of 2. Let R be the ring of integer polynomials modulo $f(x)$, $R \overset{\text{def}}{=} \mathbb{Z}[x]/(f(x))$.

Each element of R is a polynomial of degree at most $n-1$, therefore it can be associated to a coefficient vector in $\mathbb{Z}^n$: each element of R can be viewed as being both a polynomial and a vector. For $\mathbf{v}(x)$, we let $||\mathbf{v}||$ be the Euclidean norm of its coefficient vector.

Let I be an ideal of R, a subset of R that is closed under addition and multiplication by elements of R. Since I is additively closed, the coefficient vectors associated to elements of I form a lattice. I is called an ideal lattice: it is both an algebraic ideal and a lattice. Ideals have additive structure as lattices, but they also have multiplicative structure. The product IJ of two ideals I and J is the additive closure of the set $\{\mathbf{v} \times \mathbf{w} : \mathbf{v} \in I, \mathbf{w} \in J\}$, where '×' is ring multiplication.The principal ideal $(\mathbf{v})$ generated by $\mathbf{v} \in R$ corresponds to the lattice generated by the vectors $\{\mathbf{v}_i \overset{\text{def}}{=} \mathbf{v} \times x^i \bmod f(x) : i \in [0, n-1]\}$; this is called the rotation basis of the ideal lattice $(\mathbf{v})$.

Let K be a field containing the ring R; e.g. $K = \mathbb{Q}[x]/(f(x))$. The inverse of an ideal $I \subseteq R$ is $I^{-1} = \{\mathbf{w} \in K : \forall \mathbf{v} \in I, \mathbf{v} \times \mathbf{w} \in R\}$. The inverse of a principal ideal $(\mathbf{v})$ is given by $(\mathbf{v}^{-1})$, where the inverse $\mathbf{v}^{-1}$ is taken in the field K.

Lattice Problems

The main problem in lattice theory is related to short vectors. We introduce here a few fundamental problems; for more lattice problems, see [58].

- **Shortest Vector Problem (SVP):** Given a basis $B \in \mathbb{Z}^{m\times n}$, find a non-zero lattice vector $B\mathbf{x}$ (with $\mathbf{x} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$) such that $||B\mathbf{x}|| \leq ||B\mathbf{y}||$ for any other $\mathbf{y} \in \mathbb{Z}^n \backslash \{\mathbf{0}\}$.

- **Approximate SVP:** Given a lattice basis B find a non-zero lattice vector of length at most $\gamma \cdot \lambda_1(\mathcal{L}(B))$. The exact version of the problem is obtained setting the approximation factor to $\gamma = 1$, and asking for a vector of length $\lambda_1(\mathcal{L}(B))$.

- **GapSVP**$_\beta$: This problem consists of differentiating between the instances of SVP in which the answer is at most 1 or larger than β, where β can be a fixed function of n, the number of vectors. Given a basis for the lattice, the algorithm must decide whether $\lambda_1(\mathcal{L}(B)) \leq 1$ or $\lambda_1(\mathcal{L}(B)) > \beta$.

- **Closest Vector Problem (CVP):** Find a closest lattice point to a given point in the ambient space. See [58] for a more detailed description and variations.

- **Bounded distance decoding problem (BDDP):** Given a basis B of lattice Λ, a vector $\mathbf{c}$ very close to some lattice point of Λ, the aim is to find the closest point to $\mathbf{c}$ in Λ. In the promise problem

γ-BDDP, we have a parameter $\gamma > 1$ and the promise that

$$\mathrm{dist}(\Lambda, \mathbf{c}) \overset{\mathrm{def}}{=} \min_{\mathbf{v}\in\Lambda}\{||\mathbf{c}-\mathbf{v}||\} \leq \det(\Lambda)^{1/n}/\gamma$$

(BDDP is often defined with respect to λ_1 rather than with respect to $\det(\Lambda)^{1/n}$.)

- **Small Principal Ideal Problem (SPIP):** Given a principal ideal in either Hermite Normal Form or its two element representation (see Section 4.2.4), of finding a 'small' generator for it. If the SPIP is sufficiently hard, that would thwart a key recovery attack, wherein an adversary who knows the public key tries to find the secret key.

- **Polynomial Coset Problem (PCP):** [71] The problem of distinguishing between a random element of $\mathbb{Z}/d\mathbb{Z}$ and an element of the form $f(r) \bmod d$, where $f(x) \in \mathbb{Z}[x]$ is random (and unknown) with small coefficients and r is the common root of $F(x)$ and $v(x) \bmod d$ (in the notation of Section 4.2.4).

- **Sparse Subset-Sum Problem (SSSP):** Gentry, Smart-Vercauteren and Gentry-Halevi bootstrap their SHE schemes into FHE schemes using a re-encryption algorithm. Making this cryptographically secure requires an additional security assumption, namely the difficulty of a decisional version of the SSSP, i.e., it should be difficult to distinguish between random subsets of $\mathbb{Z}/d\mathbb{Z}$ and those that have sparse subsets that sum to 0. Here, bootstrapping augments the public key with a hint about the secret key, namely, with a large set of vectors that has a very sparse subset that sums to the secret key.

4.1.3 The Learning with Errors (LWE) problem

All recent lattice-based cryptographic schemes are based on one of two natural average-case problems that have worst-case hardness guarantees: the *short integer solution* (SIS) problem (which we are not going to discuss here; see [1]) and the *learning with errors* (LWE) problem.

The LWE problem was first proposed by Regev in [68], and can be described as follows. Consider a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$ uniformly at random and a noise term $e \leftarrow \chi$, and outputting $(\mathbf{a}, < \mathbf{a}, \mathbf{s} > +e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Definition 8 (LWE Problem)**.** For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ is defined as follows: given m independent samples from $A_{\mathbf{s},\chi}$ (for some $\mathbf{s} \in \mathbb{Z}_q^n$), output $\mathbf{s}$ with noticeable probability.

The (average-case) decision variant of the LWE problem, denoted $\mathsf{DLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $\mathsf{DLWE}_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{\mathbf{s},\chi}$, and is not a-priori bounded in the number of samples.

For cryptographic applications, one is primarily interested in the average case decision problem DLWE, where $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$. There are known quantum [68] and classical [67] reductions between $\mathsf{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. See also [13] for further details.

We only note here that the best known algorithms for these problems run in time nearly exponential in the dimension n [3, 60]. More generally, the best algorithms that approximate these problems to within a factor of 2^k run in time $2^{\widetilde{\mathcal{O}}(n/k)}$.

4.1.4 The Ring-Learning with Errors (RLWE) problem

Let R be a ring. Intuitively, the Ring-LWE Problem is to find s, given polynomially many $(a_i, b_i) \in R \times R$ with $b_i = a_i s + e_i$ where the a_i's are uniformly random in R, s is random in R, and the e_i's are 'small' in R. In the decisional version of Ring-LWE, one needs to distinguish such ordered pairs (a_i, b_i) from uniformly random $(a_i, u_i) \in R \times R$.

For our purposes, we describe a variant of the ring learning with errors assumption of Lyubaskevsky, Peikert and Regev [56], called polynomial LWE (or, PLWE). This is the hardness assumption used in [14]. Breaking the PLWE assumption leads to an algorithm to solve worst-case ideal lattice problems.

Definition 9 (The PLWE Assumption - Hermite Normal Form)**.** For all $k \in \mathbb{N}$, let $f(x) = f_k(x) \in \mathbb{Z}[x]$ be a polynomial of degree $n = n(k)$, let $q = q(k) \in \mathbb{Z}$ be a prime integer, let the ring $R := Z[x]/ < f(x) >$ and $R_q := R/qR$, and let χ denote a distribution over the ring R.

The polynomial LWE assumption $\mathsf{PLWE}_{f,q,\chi}$ states that for any l =poly(k) it holds that

$$\{(a_i, a_i \cdot s + e_i)\}_{i\in[l]} \overset{c}{\approx} \{(a_i, u_i)\}_{i\in[l]}$$

where s is sampled from the noise distribution χ, a_i are uniform in R_q, the "error polynomials" e_i are sampled from the error distribution χ, and finally, the ring elements u_i are uniformly random over R_q.

Notice that the PLWE assumption is defined as a decisional assumption. One could also define the search assumption which requires an adversary to find $s \in R_q$, given any polynomial number of samples $(a_i, a_i \cdot s + e_i)$. The search and decisional assumptions are equivalent for some range of parameters, as shown in [56].

4.1.5 The General Learning with Errors (GLWE) Problem

For completeness, following [12] we report here the GLWE Problem. The LWE and the RLWE problems are syntactically identical, aside from using different rings ($\mathbb{Z}$ versus a polynomial ring) and different vector dimensions over those rings ($n = \text{poly}(\lambda)$ for LWE, but n is constant - namely, 1 - in the RLWE case). A General Learning with Errors (GLWE) Problem is introduced to simplify discussion and describe a single GLWE- based FHE scheme, rather than presenting essentially the same scheme twice, once for each of the two concrete instantiations. (This will be used in the scheme [12].)

Definition 10 (GLWE)**.** For security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2, let $q = q(\lambda) \geq 2$ be a prime integer, let $R = \mathbb{Z}[x] = (f(x))$ and $R_q = R/qR$, and let $\chi = \chi(\lambda)$ be a distribution over R. The $\mathsf{GLWE}_{n,f,q,\chi}$ problem is to distinguish the following two distributions:

- one samples $(\mathbf{a}_i, b_i)$ uniformly from R_q^{n+1}.
- one first draws $\mathbf{s} \leftarrow R_q^n$ uniformly and then samples $(\mathbf{a}_i, b_i) \in R_q^{n+1}$ by sampling $\mathbf{a}_i \leftarrow R_q^n$ uniformly, $e_i \leftarrow \chi$, and setting $b_i =< \mathbf{a}_i, \mathbf{s} > +e_i$.

The $\mathsf{GLWE}_{n,f,q,\chi}$ assumption is that the $\mathsf{GLWE}_{n,f,q,\chi}$ problem is infeasible.

LWE is simply GLWE instantiated with $d = 1$. RLWE is GLWE instantiated with $n = 1$. Interestingly, as far as we know, instances of GLWE between these extremes have not been explored. One would suspect that GLWE is hard for any (n, d) such that $n \cdot d = \Omega(\lambda \log (q/B))$, where B is a bound (with overwhelming probability) on the length of elements output by χ. For fixed $n \cdot d$, perhaps GLWE gradually becomes harder as n increases (if it is true that general lattice problems are harder than ideal lattice problems), whereas increasing d is probably often preferable for efficiency.

The GLWE assumption implies that the distribution $\{(\mathbf{a}_i, < \mathbf{a}_i, \mathbf{s} > +t \cdot e_i)\}$ is computational indistinguishable from uniform for any t relatively prime to q. This fact is convenient for encryption, where, for example, a message m may be encrypted as $(\mathbf{a}, < \mathbf{a}, \mathbf{s} > +2e + m)$, and this fact can be used to argue that the second component of this message is indistinguishable from random.

4.1.6 NTRU

NTRU is a patented and open source public-key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data. It consists of two algorithms: NTRUEncrypt, which is used for encryption, and NTRUSign, which is used for digital signatures. Unlike other popular public-key cryptosystems, it is resistant to attacks using Shor's algorithm and its performance has been shown to be significantly better.

The first version of the system, which was called NTRU, was developed in 1996 by J. Hoffstein, J. Pipher, and J.H. Silverman [48]. That same year, the developers of NTRU joined with D. Lieman and founded the NTRU Cryptosystems, Inc., and were given a patent on the cryptosystem. In 2013, D. Stehle and R. Steinfeld created [74] a provably secure version of NTRU [48] which is being studied by a post quantum crypto group chartered by the European Commission. We describe here this version from [74].

For a security parameter λ, the scheme is parametrized by the following quantities:

- an integer $n = n(\lambda)$,
- a prime number $q = q(\lambda)$,
- a degree-n polynomial $\phi(x) = \phi_\lambda(x)$,
- a $B(\lambda)$-bounded error distribution $\chi = \chi(\lambda)$ over the ring $R \overset{\text{def}}{=} \mathbb{Z}[x]/(\phi(x))$.

The parameters $n, q, \phi(x)$ and χ are public and we assume that given λ, there are polynomial-time algorithms that output n, q and $\phi(x)$, and sample from the error distribution χ. The message space is $\mathcal{M} = \{0, 1\}$, and all operations on ciphertexts are carried out in the ring R_q (i.e. modulo q and $\phi(x)$). The algorithms of the encryption schemes are as follows.

- $\mathsf{Keygen}(\lambda)$: Sample polynomials $f', g \leftarrow \chi$ and set $f := 2f' + 1$ so that $f \equiv 1 \bmod 2$. If f is not invertible in R_q, resample f'. Set

$$\mathsf{pk} := h = 2gf^{-1} \in R_q, \quad \mathsf{sk} := f \in R$$

- $\mathsf{Enc}(\mathsf{pk}, m)$: Sample polynomials $s, e \leftarrow \chi$. Output the ciphertext

$$c := hs + 2e + m \in R_q$$

where all operations are done modulo q and $\phi(x)$.

- $\mathsf{Dec}(\mathsf{sk}, c)$: Let $\mu = fc \in R_q$. Output $m' := \mu \bmod 2$.

It is easily seen that this scheme is correct as long as there is no wrap-around modulo q. To decrypt a ciphertext c, we compute in R_q:

$$fc = fhs + 2fe + fm = 2gs + 2fe + fm$$

If there is no wrap-around modulo q then

$$fc \bmod 2 = 2gs + 2fe + fm \bmod 2 = fm \bmod 2 = m$$

One possible setting which ensures that there is no wrap-around modulo q is to set $\phi(x) = x^n + 1$. To see why this helps, notice that since the coefficients of g, s, f, e are all bounded by $2B + 1$ (the coefficients of g, s and e are bounded by B and that of f is bounded by $2B + 1$). We can say that the coefficients of $gs \pmod{x^n + 1}$ and $fe \pmod{x^n + 1}$ are both bounded by $n(2B + 1)^2$. Thus, the coefficients of fc are bounded by $4n(2B + 1)^2 + B < 36nB^2 < q/2$. In other words, as long as we set $q > 72nB^2$, a fresh ciphertext generated by Enc is guaranteed to decrypt correctly. From here on, we refer to $\mu = fc \in R_q$ as the *error* in a ciphertext c.

For more details and for the security analysis of this scheme, we refer to [74] or [55].

Remark 1. D. Bernstein, T. Lange et al. released in 2016 NTRU Prime [7] which adds defenses against potential attacks to NTRU by eliminating some worrisome algebraic structure. At equivalent cryptographic strength, we know that NTRU performs costly operations on the private key much faster than RSA does. The time of performing an RSA private operation increases as the cube of the key size, whereas that of an NTRU operation increases in a quadratic way.[3]

4.2 Lattice Based FHE Schemes

Following exposition in [40], we are going now to describe shortly Gentry's original scheme [37], and then we will proceed describing a few variations made by Smart-Vercauteren [71] and Gentry-Halevi [40].

4.2.1 First Approach: Lattice-based Cryptosystems

We start by describing a typical construction for a lattice-based encryption scheme, the GGH lattice-based cryptosystem [45] in the improved version of Micciancio [57]; Gentry's scheme is based on the same idea. Given a lattice Λ, the secret and public keys are a ”good” and a ”bad” base (respectively) of Λ. Namely, a good basis B_{sk} consists of short, ”nearly orthogonal” vectors, while the public key is the HNF of Λ, $B_{\text{pk}} \overset{\text{def}}{=} \text{HNF}(\mathcal{L}(B_{\text{sk}}))$. (See Figure 4.2.)

[3] There are other reasons for considering NTRU cryptography as a good candidate for post-quantum cryptography. Unlike RSA and Elliptic Curve Cryptography, NTRU is not known to be vulnerable to quantum computer based attacks. The National Institute of Standards and Technology wrote in a 2009 survey that among the viable alternatives for both public key encryption and signatures that are not vulnerable to Shor's Algorithm, the NTRU family of cryptographic algorithms appears to be the most practical. The European Union's PQCRYPTO project (Horizon 2020 ICT-645622) is evaluating the provably secure Stehle–Steinfeld version of NTRU (not original NTRU algorithm itself) as a potential European standard. However the Stehle-Steinfeld version [74] of NTRU is significantly less efficient than the original scheme.

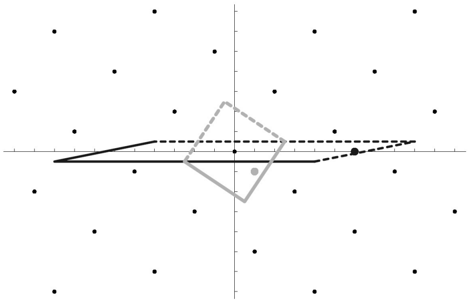


Fig. 4.2 Good and bad bases

A ciphertext in a GGH-type cryptosystem is a vector $\mathbf{c}$ close to the lattice $\mathcal{L}(B_{\text{pk}})$, and plaintext message is embedded in the distance from $\mathbf{c}$ to the closest lattice vector. To encrypt a message m, the sender chooses a short error vector $\mathbf{e}$ that encodes m, and then computes the ciphertext as $\mathbf{c} \leftarrow \mathbf{e} \bmod B_{\text{pk}}$. To decrypt, one recovers $\mathbf{e}$ with $\mathbf{e} \leftarrow \mathbf{c} \bmod B_{\text{sk}}$, and then recovers m from $\mathbf{e}$.

The idea behind the correctness is that the fundamental parallelepiped $\mathcal{P}(B_{\text{sk}})$ related to the secret key is a "full rounded shape" parallelepiped that contains a sphere of radius bigger than $||\mathbf{e}||$, so that $\mathbf{e}$ is the point inside $\mathcal{P}(B_{\text{sk}})$ that equals $\mathbf{c}$ modulo Λ. On the other hand, the parallelepiped $\mathcal{P}(B_{\text{pk}})$ related to the public key is very skewed, and does not contain a sphere of large radius, making it useless for solving BDDP.

4.2.2 Gentry's Scheme

We are going to present the scheme at a high level; see the original paper [37] for details. Gentry's SHE scheme can be seen as a GGH-type scheme over ideal lattices. The public key consists of a bad basis B_{pk} of an ideal lattice J, along with some basis B_I of a "small" ideal I (which is used to embed messages into the error vectors). For example, the small ideal I can be taken to be $I = (2)$, the set of vectors with all even coefficients.

A ciphertext in Gentry's scheme is a vector close to a J-point, with the message being embedded in the distance to the nearest lattice point. More specifically, the plaintext space is $\{0, 1\}$, which is embedded in $R/I = \{0, 1\}^n$ by encoding 0 as 0^n and 1 as $0^{n-1}1$. For an encoded bit $\mathbf{m} \in \{0, 1\}^n$ we set $\mathbf{e} = 2\mathbf{r} + \mathbf{m}$ for a random small vector $\mathbf{r}$, and then output the ciphertext $\mathbf{c} \leftarrow \mathbf{e} \bmod B_{\text{pk}}$.

The secret key in Gentry's scheme (that plays the role of the 'good basis' of J) is just a short vector $\mathbf{w} \in J^{-1}$. Decryption involves computing the fractional part $[\mathbf{w} \times \mathbf{c}]$. Since $\mathbf{c} = \mathbf{j} + \mathbf{e}$ for some $\mathbf{j} \in J$, then $\mathbf{w} \times \mathbf{c} = \mathbf{w} \times \mathbf{j} + \mathbf{w} \times \mathbf{e}$. But $\mathbf{w} \times \mathbf{j}$ is in R and thus an integer vector, so $\mathbf{w} \times \mathbf{c}$ and $\mathbf{w} \times \mathbf{e}$ have the same fractional part, $[\mathbf{w} \times \mathbf{c}] = [\mathbf{w} \times \mathbf{e}]$. If $\mathbf{w}$ and $\mathbf{e}$ are short enough - in particular, if we have the guarantee that all of the coefficients of $\mathbf{w} \times \mathbf{e}$ have magnitude less than $1/2$ - then $[\mathbf{w} \times \mathbf{e}]$ equals $\mathbf{w} \times \mathbf{e}$ exactly. From $\mathbf{w} \times \mathbf{e}$, the decryptor can multiply by $\mathbf{w}^{-1}$ to recover $\mathbf{e}$, and then recover $\mathbf{m} \leftarrow \mathbf{e} \bmod 2$. The actual decryption procedure from [37] is slightly different, however. Specifically, $\mathbf{w}$ is "tweaked" so that decryption can be implemented as $\mathbf{m} \leftarrow \mathbf{c} - [\mathbf{w} \times \mathbf{c}] \bmod 2$ (when $I = (2)$).

4.2.3 Gentry's Fully Homomorphic Scheme

The scheme is SHE: given two ciphertexts $\mathbf{c}_1 = \mathbf{j}_1 + \mathbf{e}_1$ and $\mathbf{c}_2 = \mathbf{j}_2 + \mathbf{e}_2$, their sum is $\mathbf{j}_3 + \mathbf{e}_3$ where $\mathbf{j}_3 = \mathbf{j}_1 + \mathbf{j}_2 \in J$ and $\mathbf{e}_3 = \mathbf{e}_1 + \mathbf{e}_2$ is small. Similarly, their product is $\mathbf{j}_4 + \mathbf{e}_4$ where $\mathbf{j}_4 = \mathbf{j}_1 \times (\mathbf{j}_2 + \mathbf{e}_2) + \mathbf{e}_1 \times \mathbf{j}_2 \in J$ and $\mathbf{e}_4 = \mathbf{e}_1 \times \mathbf{e}_2$ is still small. Gentry's SHE scheme can evaluate low-degree polynomials but not more. Once the degree (or the number of terms) is too large, the error vector $\mathbf{e}$ grows beyond the decryption capability of the private key. Gentry solved this problem using bootstrapping: in [37] he showed that a scheme that can homomorphically evaluate its own decryption circuit plus one additional operation, can be transformed into a fully-homomorphic encryption. In more details, fix two ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ and consider the functions

$$\begin{aligned}\mathrm{DAdd}_{\mathbf{c}_1,\mathbf{c}_2}(\mathrm{sk}) &\overset{\mathrm{def}}{=} \mathrm{Dec}_{\mathrm{sk}}(\mathbf{c}_1) + \mathrm{Dec}_{\mathrm{sk}}(\mathbf{c}_2)\\ \mathrm{DMul}_{\mathbf{c}_1,\mathbf{c}_2}(\mathrm{sk}) &\overset{\mathrm{def}}{=} \mathrm{Dec}_{\mathrm{sk}}(\mathbf{c}_1) \times \mathrm{Dec}_{\mathrm{sk}}(\mathbf{c}_2)\end{aligned}$$

We said earlier that a SHE scheme is bootstrappable if it is capable of homomorphically evaluating the functions $\mathrm{DAdd}_{\mathbf{c}_1,\mathbf{c}_2}$ and $\mathrm{DMul}_{\mathbf{c}_1,\mathbf{c}_2}$ for any two ciphertexts $\mathbf{c}_1, \mathbf{c}_2$. Given a bootstrappable scheme that is also circular secure, it can be transformed into a fully-homomorphic scheme by adding to the public key an encryption of the secret key, $\mathbf{c}^* \leftarrow \mathrm{Enc}_{\mathrm{pk}}(\mathrm{sk})$. Then given any two ciphertexts $\mathbf{c}_1, \mathbf{c}_2$, the addition/multiplication of these two ciphertexts can be computed by homomorphically evaluating the functions $\mathrm{DAdd}_{\mathbf{c}_1,\mathbf{c}_2}(\mathbf{c}^*)$ or $\mathrm{DMul}_{\mathbf{c}_1,\mathbf{c}_2}(\mathbf{c}^*)$.

This SHE scheme is not bootstrappable. Gentry showed how to squash the decryption circuit, transforming the original SHE scheme E into a scheme E^* that can correctly evaluate any circuit that E can, but where the complexity of E^*'s decryption circuit is much less than E's. In the original SHE scheme E, the secret key is a vector $\mathbf{w}$. In the new scheme E^*, the public key includes an additional 'hint' about $\mathbf{w}$ - namely, a big set of vectors $\mathcal{S} = \{\mathbf{x}_i : i = 1, 2, \ldots, S\}$ that have a hidden sparse subset T that adds up to $\mathbf{w}$. The secret key of E^* is the characteristic vector of the sparse subset T, which is denoted $\sigma =< \sigma_1, \sigma_2, \ldots, \sigma_S >$.

Whereas decryption in the original scheme involved computing $\mathbf{m} \leftarrow \mathbf{c} - [\mathbf{w} \times \mathbf{c}] \bmod 2$, in the new scheme the ciphertext $\mathbf{c}$ is 'post-processed' by computing the products $\mathbf{y}_i = \mathbf{x}_i \times \mathbf{c}$ for all of the vectors $\mathbf{x}_i \in \mathcal{S}$. The decryption in the new scheme can be done by computing $\mathbf{c} - [\sum_j \sigma_j \mathbf{y}_j] \bmod 2$, and this computation can be expressed as a polynomial in the σ_i's of degree roughly the size of the sparse subset T. With appropriate setting of the parameters, the subset T can be made small enough to get a bootstrappable scheme.

4.2.4 Variations of Gentry's Scheme

We describe now two variations on Gentry's SHE schemes. These are

- Smart and Vercauteren in [71]
- Gentry and Halevi in [40]

The [71] scheme has smaller message expansion and key size than Gentry's original scheme. The authors implemented the SHE scheme, but were not able to implement the bootstrapping functionality. The authors of [40] improved the scheme with, among others, a better key-generation method for SHE scheme which reduces the asymptotic complexity from $\widetilde{O}(n^{5/2})$ to $\widetilde{O}(n^{3/2})$ when working with dimension-n lattices (and practically reducing the time from many hours/days to a few seconds/minutes). This allowed them to implement all aspects of the scheme, including the bootstrapping functionality.

4.2.5 [SV10] and [GH11b] Variations

To simplify the presentation we present the two versions as schemes that encrypt elements in the message space $\mathbb{F}_2 = \{0, 1\}$. Let also, for a positive value t and integer N,

$$\begin{aligned} \mathcal{B}_{\infty,N}(t) &:= \left\{ \sum_{i=0}^{N-1} a_i x^i : -t \leq a_i \leq t, a_i \in \mathbb{Z} \right\} \\ &= \{a(x) \in \mathbb{Z}[x] \text{ s.t. } ||a(x)||_\infty \leq t\} \end{aligned}$$

We will consider parameters (N, η, μ). Typically, $N = 2^n$ for some dimension n, $\eta = 2^{\sqrt{N}}$ and μ small integer, for instance $\mu = 1$ (or in any case $\mu \leq \sqrt{N}$).

SV Scheme

$\mathsf{KeyGen}(1^\lambda)$

- choose a monic irreducible polynomial $F(x) \in \mathbb{Z}[x]$ of degree N
- **repeat**:
  - $S(x) \xleftarrow{\$} \mathcal{B}_{\infty,N}(\eta/2)$
  - $v(x) \leftarrow 1 + 2 \cdot S(x)$
  - $d \leftarrow \mathsf{resultant}(v(x), F(x))$
- **until** d is prime. Let $p := d$.
- $D(x)$ ←gcd$(v(x), F(x))$ over $\mathbb{F}_p[x]$
- let $r \in \mathbb{F}_p$ denote the unique root of $D(x)$
- compute $w(x) = \sum_{i=0}^{N-1} w_i x^i \in \mathbb{Z}[x]$ s.t.

  $$w(x) \cdot v(x) = p \bmod F(x)$$

- $B \leftarrow w_0 \bmod 2p$
- $\mathsf{pk} \leftarrow (p, r, \mu)$
- $\mathsf{sk} \leftarrow (p, B)$

$\mathsf{Encrypt}(m, \mathsf{pk})$

- $u(x) \xleftarrow{\$} \mathcal{B}_{\infty,N}(\mu/2)$
- $a(x) \leftarrow m + 2 \cdot u(x)$
- $c \leftarrow [a(r)]_p$

$\mathsf{Decrypt}(c, \mathsf{sk})$

- $m \leftarrow (c - \lfloor c \cdot B/p \rceil) \bmod 2$

GH Scheme

$\mathsf{KeyGen}(1^\lambda)$

- choose a monic irreducible polynomial $F(x) := x^N + 1 \in \mathbb{Z}[x]$
- **repeat**:
  - pick $v(x) = \sum_{i=0}^{N-1} v_i x^i \xleftarrow{\$} \mathcal{B}_{\infty,N}(\eta/2)$
  - consider the matrix V generated by $v(x)$ as given in (4.1)
  - let d =det(V) = $\mathrm{Resultant}(v(x), F(x))$
- **until** $v(x)$ is a good generating polynomial. (We consider $v(x)$ to be good if HNF(V) has the same form as in (4.2). It was observed by N. Smart that this happens if d is odd and square-free.)
- let r be the unique common root of $F(x)$ and $v(x)$ modulo d
- compute the polynomial $w(x) = \sum_{i=0}^{N-1} w_i x^i \in \mathbb{Z}[x]$ such that $w(x) \cdot v(x) = d \bmod F(x)$
- $\mathsf{sk} \leftarrow w$, where w is one of odd coefficients of $w(x)$
- $\mathsf{pk} \leftarrow \{r, d, \mu\}$

$\mathsf{Encrypt}(m, \mathsf{pk}, \mu)$

- generate a random noise vector $\mathbf{u} := (u_0, u_1, \ldots, u_{N-1})$, where $u_i \leq \mu$
- set $\mathbf{a} := 2\mathbf{u} + m \cdot \mathbf{e}_1$
- let $\mathbf{a} = (a_0, a_1, \ldots, a_{N-1})$
- let $a(x) := \sum_{i=0}^{N-1} a_i x^i$
- let the ciphertext be the integer $c := [a(r)]_d = [m + 2\sum_{i=1}^{N-1} u_i r^i]_d$

$\mathsf{Decrypt}(c, \mathsf{sk})$

- $m \leftarrow [c \cdot w]_d \bmod 2$

For a vector $\mathbf{v} := (v_0, \ldots, v_{N-1})$, consider the matrix

$$V := \begin{pmatrix} v_0 & v_1 & \cdots & v_{N-1} \\ -v_{N-1} & v_0 & \cdots & v_{N-2} \\ & & \cdots & \\ -v_1 & -v_2 & \cdots & v_0 \end{pmatrix} \in \mathcal{M}_N(\mathbb{Z}) \tag{4.1}$$

Also, for d the unique common root of $F(x)$ and $v(x)$, letting $r_i = r^i \bmod d$, the HNF of the ideal lattice $J = (\mathbf{v})$ is

$$B = \mathrm{HNF}(J) := \begin{pmatrix} d & 0 & 0 & \cdots & 0 \\ -r_1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -r_{N-1} & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathcal{M}_N(\mathbb{Z}) \tag{4.2}$$

We remark that in [39], Gentry and Halevi build a bootstrappable SHE by replacing the SSSP hardness assumption with the decision Diffie-Hellman. Therefore, they achieve a leveled FHE scheme without the squashing step; moreover, exactly as in Gentry's blueprint, a pure FHE is achieved by assuming circular security.

The authors of [71] also show how to obtain a FHE scheme from their SHE scheme. See the original paper for further details.

Security

Gama and Nguyen performed several experiments with lattices in dimensions 100-400 [34]. They concluded that for those dimensions it is feasible to solve γ-BDDP when $\gamma > 1.01^n \approx 2^{n/70}$. The best algorithms for solving the γ-BDDP in n-dimensional lattices take time exponential in $n/\log\gamma$:

currently known algorithms can solve dimension-n γ-BDDP in time 2^k up to $\gamma = 2^{\frac{\mu n}{k/\log k}}$, where μ is a parameter that depends on the exact details of the algorithm. (Extrapolating from the Gama-Nguyen experiments, $\mu \in [0.1, 0.2]$.)

The primary known attacks on FHE schemes are variants of the LLL lattice basis reduction algorithm [52]. The security of almost all currently known schemes is based on the presumed difficulty of some lattice problem, such as finding an approximately shortest (non-zero) vector in a high dimensional lattice.

As we saw, a number of FHE schemes use ideal lattices rather than arbitrary lattices. Since these are very special lattices, it might turn out to be the case that lattice attacks are easier for ideal lattices than for generic lattices. This is an open question. At the moment, special attacks that work better for ideal lattices than for general lattices are not yet known.

The security of these schemes is based on the simultaneous difficulty of problems like SPIP, PCP, or SSSP.

Chapter 5

Key-Recovery Attacks against Existing SHE Schemes

We present in this chapter the main contribution of our dissertation. In Chapter 4 we learned under which hardness assumptions we divide the known SHE schemes, and in Chapter 1 we saw what can be the impact of a key-recovery attack on a SHE scheme that can be used in a real-case scenario. We therefore describe here our key-recovery attacks.

Definition 11. Consider a scenario in which an attacker is attempting a non-adaptive chosen-ciphertext attack to a given SHE scheme. We say that we can perform a successful *key-recovery attack* if we can completely determine all the bits of the secret key sk.

5.1 Related Work

Our starting point is the work of [54], whose authors were the first to observe key recovery attacks by showing that the SHE schemes falling in the category (1) in Figure 4.1is vulnerable to such attacks.

5.1.1 Initial Attempt

In 2011, Loftus, May, Smart and Vercauteren [54] showed that the basic Gentry scheme [37] is not IND-CCA1; indeed a lunchtime attack allows one to recover the secret key. They then showed that a minor modification to the variant of the SHE scheme [71] of Smart and Vercauteren will allow one to achieve IND-CCA1, indeed PA-1, in the standard model assuming a lattice based knowledge assumption.

We informally present the attack; for the details of the attack, see [54]. The decryption algorithm is $m \leftarrow [c \cdot w]_d \bmod 2$. This decryption will be valid as long as $c \cdot w/d \leq 1/2$. Therefore, for a certain key set (w, d), the maximum value c' allowed is a fixed integer. The adversary picks several different 'ciphertexts', and pass them to the decryption oracle to check if they can be decrypted correctly. Eventually, the attacker will recover the threshold c' which is the maximum integer that can be decrypted correctly. Then it is easy for the attacker to recover w, the secret key.

Recovering the secret key will require $\mathcal{O}(\log\, d)$ calls to the oracle. The attack is highly efficient in practice and recovers keys in a matter of seconds for all parameter sizes in [40].

To stop this attack, in [54] the authors proposed a ciphertext check procedure. The ciphertext that is to be decrypted, will be disassembled into the generating polynomial $a(x)$. Recall that $a(x) = m + 2u(x)$, hence, for valid ciphertexts, $||a(x)||_\infty$ is bounded by a certain threshold smaller than T - where $T > \frac{1}{4}||a(x)||_\infty$ - while for invalid 'ciphertexts' (i.e., integers picked by attacker), the corresponding $a(x)$ can have arbitrary coefficients. Therefore, in the latter case, an error $\perp$ is generated, and the decryption stops. For the details of the ciphertext-checking SHE (ccSHE) scheme, see [54, p. 8].

The authors show that the ccSHE scheme is PA-1 secure assuming a particular lattice knowledge assumption (see [54]). This is the only IND-CCA1 secure SHE scheme known so far; but it makes use of a non-standard lattice knowledge assumption.

5.1.2 Follow-Up Work

Zhang, Plantard and Susilo showed [76] an IND-CCA1 attack against the SHE scheme [75], therefore showing that family in (2) is not IND-CCA1 secure. Given $O(\lambda^2)$ decryption oracle queries, an attacker can recover the private key. Let η be the bit-length of the secret key p, $O(\lambda^2) = 3(\eta + 3)$ in the best case. We will describe a more efficient and conceptually simpler key recovery attack. Our attack is optimal in the sense that it recovers directly the secret key with at most η oracle queries.

Loftus, May, Smart, and Vercauteren showed in [54] that the schemes in the family (1) in Figure 4.1 are not IND-CCA1 secure. With our work, we show that the remaining families (2), (3) and (4) are also not IND-CCA1 secure.

Recently, Galbraith et al. [53] explore a new approach to achieving security against adaptive attacks, which does not rely on a notion of ”valid ciphertexts” (see also Remark 2). The idea is to generate a ”one-time” private key every time the decryption algorithm is run, so that even if an attacker can learn some bits of the one-time private key from each decryption query, this does not allow them to compute a valid private key. They show an implementation of their idea on the [44] SHE scheme.

5.2 Framework of Key Recovery Attacks

This is the general framework of how our key-recovery attacks work. First of all, notice that most of the SHE schemes considered work by encrypting only one bit at a time; therefore, also the decryption will reveal in general only one bit. We can therefore see that for every ciphertext submitted to the decryption oracle, this will return us the corresponding plaintext bit. Now, the idea is that we are not necessarily forced to submit to the decryption oracle a correctly-generated ciphertext; without proper ciphertext-validity check (see Remark 2), we can actually submit to the decryption oracle any value picked from the ciphertext domain (namely, if $\mathcal{C} \subseteq C$ is the set of all possible ciphertexts, we can choose and submit to the decryption oracle any value $c' \in C$). The decryption oracle will compute and return the value $d = \mathsf{Decrypt}(\mathsf{sk}, c') \in M$, for a known set of 'plaintexts' M. The key idea here is that, whatever value $c' \in C$ we are submitting to the decryption oracle, it will return a value $d \in M$ (normally a single bit) *which is a function of the secret key* sk. Therefore the idea is to submit to the decryption oracle specifically-chosen 'ciphertexts' in order to obtain, after every single query, a different bit of the secret key sk. By studying how the decryption step works we can repeat and vary our queries and finally recover completely all the bits of the secret key.

Remark 2. One can argue that it is enough to provide the decryption step in a SHE scheme with an extra "ciphertext-validity check" in order to thwart this specifically-chosen ciphertext approach. However, it can be seen that implementing such an extra check is not easy task. After attacking the SHE schemes as we will see in this chapter, we tried to device extra validity checks but we also found out that we can always modify slightly our attacks in order to prevent these extra checks to take effect. The only exception is given by a modification of the [55] SHE scheme which we developed, and that so far seems to resist key-recovery attacks. This gives us good indicator that our variation of the SHE scheme [55] is actually resistant to key-recovery attacks, even though further investigation is needed. We will come back to this topic in Chapter 6.

Often in the following discussion we will say that our key-attacks are optimal. Let's explain what do we mean with this. Let u be the number of bits revealed by the decryption oracle after each query, and let N be the number of bits of the secret key sk. Then, assuming that the bits of the secret key can be considered - at least from the attacker's perspective - uniformally chosen at random, the minimum number of decryption queries to perform in order to obtain all the bits of the secret key is $v_{\min} := N/u$.

Definition 12. Consider a SHE scheme for which we have devised a key-recovery attack. Let v be the number of oracle decryption queries that we have to perform in order to fully recover the secret key. We say that our key-recovery attack is *optimal* for this SHE scheme if $v = v_{\min}$.

It is often the case that $u = 1$ and therefore $v = N$ or $v \approx N$.

Remark 3. We can also notice that one does not need to recover completely *all* the bits of the secret key; if our task is to beat the IND-CCA1 security, then it may be enough to recover only a given number of bit of sk in order to have a working CCA1 attack. (Which, we remember, is weaker than a key-recovery attack.) Precisely, for a given SHE scheme consider the secret key $\mathsf{sk} = s$. Let $N = \#\text{bits}(s)$. Let $0 \leq n \leq N$ the number of bits of s that we have learned through a given number of oracle decryption queries. Then obviously we have that

- if $n = 0$ we have no information whatsoever about s, and therefore can cannot beat CCA1 security;

- if $n = N$ then we have completely recovered s, and in particular we can beat CCA1 security;

The interesting situation is when $0 < n < N$. There is a value $t := f(s) \in \mathbb{N}$ which is function of s such that

- if $n < t$ then we cannot beat CCA1 security;

- if $n > t$ then we can beat CCA1 security;

This is material for an interesting future work. We will come back on this topic in Section 7.1.

5.2.1 The Framework of Key-Recovery Attacks

According to the given SHE scheme to attack, we have devised two main approaches:

- for SHE schemes like [75, 11, 55, 9], we recover sk by gradually reducing (halving) the key space

- for SHE schemes like [15, 14, 12, 44], we recover sk bit-by-bit, from least to most significant bit

As we already mentioned, in general our attacks are optimal, in the sense of definition 12.

In this chapter we will describe all the Key-Recovery attacks that we have developed against the SHE schemes in categories (2), (3) and (4) as in Figure 4.1; more precisely against the SHE schemes [75, 15, 14, 12, 11, 44, 55, 9].

5.3 Key Recovery Attack against the DGHV10 Scheme

In [76], Zhang, Plantard and Susilo presented a key recovery attack against the SHE scheme [75]. Given $O(\lambda^2)$ decryption oracle queries, an attacker can recover the private key. Let η be the bit-length of the secret key p, $O(\lambda^2) = 3(\eta + 3)$ in the best case.

In this section, we describe a more efficient and conceptually simpler key recovery attack. Our attack is optimal in the sense that it recovers directly the secret key with at most η oracle queries. Note that the decryption oracle outputs one bit at a time.

5.3.1 The DGHV10 SHE Scheme

We start by presenting the (asymmetric) SHE scheme as presented in [75]. For this SHE scheme - as well for the ones we will describe subsequently - we will omit the evaluation step; for a complete description of the scheme, see the original papers. The FHE scheme from [75] is perhaps the simplest and easiest to understand since it relies only on simple arithmetics.The message space is $\mathcal{M} = \mathbb{Z}_2$. The scheme is parametrized by γ (bit-length of the integers in the public key), η (bit-length of the secret key), ρ (bit-length of the noise), and τ (the number of integers in the public key). We also consider a secondary noise parameter $\rho' = \rho + \omega(\log\lambda)$. For a specific ($\eta$-bit) odd positive integer p, consider the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \{\text{choose } q \overset{\$}{\leftarrow} \mathbb{Z}\cap[0, 2^\gamma/p), r \overset{\$}{\leftarrow} \mathbb{Z}\cap(-2^\rho, 2^\rho) : \text{output } x = pq+r\}$$

The algorithms of the DGHV10 SHE scheme are defined as follows:

$\mathsf{KeyGen}(\lambda)$

- sk: odd η-bit integer
  $p \xleftarrow{\$} (2\mathbb{Z}+1) \cap [2^{\eta-1}, 2^{\eta})$.
- sample $x_i \xleftarrow{\$} \mathcal{D}_{\gamma,\rho}(p)$ for $i = 0, \ldots, \tau$
- relabel so that x_0 is the largest
- restart unless x_0 odd, $r_p(x_0)$ even ($r_p(x) = x - \lfloor x/p \rceil \cdot p \in (-p/2, p/2]$)
- $\mathsf{pk} = (x_0, x_1, \ldots, x_\tau)$.

$\mathsf{Encrypt}(\mathsf{pk}, m \in \mathcal{M})$

- choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$
- choose a random integer r in $(-2^{\rho'}, 2^{\rho'})$
- output $c = [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$

$\mathsf{Decrypt}(\mathsf{sk}, c)$

- output $m' = (c \bmod p) \bmod 2$

5.3.2 The New Key Recovery Attack

Since $\eta = \#\text{bits}(p)$, we immediately obtain odd lower and upper bounds l_p and u_p, respectively, for p:

$$l_p = 2^{\eta-1} + 1 \le p \le u_p = 2^{\eta} - 1$$

Notice explicitly that p can only assume the odd values $2^{\eta-1}+1, 2^{\eta-1}+3, \ldots, 2^{\eta}-3, 2^{\eta}-1$. In particular, between $2^{\eta-1}$ and 2^{η} there are $2^{\eta-2}$

candidate values for p. We can also argue that between l_p and u_p there are $(u_p - l_p)/2 = 2^{\eta-2} - 1$ even integers. Let $H_{(l_p,u_p)} = \{0, 1, \ldots, 2^{\eta-2} - 2\}$, these integers can be denoted as $l_p + 2h + 1$ for $h \in H_{(l_p,u_p)}$.

Now, the idea of the key-recovery attack is as follows: consider the 'ciphertext' $c = l_p + 2h + 1$ for a given $h \in H_{(l_p,u_p)}$. Submit c to the decryption oracle $\mathcal{O}_D$; we will obtain a bit $b \leftarrow \mathcal{O}_D(c) = (c \bmod p) \bmod 2$. There are two cases to distinguish:

$\mathbf{b = 0}$ 'Decryption is correct' (since c is even); hence $p > c$, i.e. $p \geq l_p + 2h + 2$.

Update $l_p \leftarrow l_p + 2h + 2$.

$\mathbf{b = 1}$ 'Decryption is not correct'; hence $p < c$, i.e. $p \leq l_p + 2h$.

Update $u_p \leftarrow l_p + 2h$.

Next, we repeat the decryption query with the updated values for l_p, u_p and with another even 'ciphertext' $c \in [l_p + 1, u_p - 1]$, and we stop when $u_p = l_p$. In particular, for efficiency we always choose c as the even integer in the middle of the interval $[l_p + 1, u_p - 1]$. It is easy to see that this attack leads to a full recovery of the secret key p with at most $\log(2^{\eta-2} - 2) \approx \eta$ oracle queries.

5.3.3 Algorithmic Description

Formally, the attack can be described by Algorithm 1. It takes as input an integer $\eta \in \mathbb{N}$ and outputs the secret integer p. Let

$$\begin{aligned} \mathcal{O}_D(c) &= \mathsf{Decrypt}(c, sk) = (c \bmod p) \bmod 2 \quad \text{(decryption oracle)} \\ \lfloor x \rfloor_{\mathrm{o}} &= \max\{n \in \mathbb{N} \text{ s.t. } n \text{ is odd and } n \leq x\} \end{aligned}$$

Algorithm 1 Key Recovery Attack

```
input: $\eta$
$l_p \leftarrow 2^{\eta-1} + 1$
$u_p \leftarrow 2^{\eta} - 1$
while $u_p \neq l_p$ do
  $h \leftarrow (u_p - l_p)/2$ {$h \in \mathbb{N}$ is the number of even values in $[l_p, u_p]$}
  $c \leftarrow l_p + \lfloor h \rfloor_o$
  if $\mathcal{O}_D(c) = 0$ then
    $l_p \leftarrow l_p + \lfloor h \rfloor_o + 1$
  end if
  if $\mathcal{O}_D(c) = 1$ then
    $u_p \leftarrow l_p + \lfloor h \rfloor_o - 1$
  end if
end while
return $u_p$
```

5.4 Key Recovery Attack against the BV11b Scheme

In this section, we describe a key recovery attack against the SHE scheme from [15].

5.4.1 The BV11b SHE Scheme

The message space is $\mathcal{M} = \mathbb{Z}_2$. Let f be a polynomial in λ, i.e. $f(\lambda) = \text{poly}(\lambda)$. Consider $n = f(\lambda) \in \mathbb{N}$ and let $\epsilon \in (0, 1) \cap \mathbb{R}$. Assume an odd integer $q \in \mathbb{N}$ such that $q \in [2^{n^\epsilon}, 2 \cdot 2^{n^\epsilon})$, and an integer $m \geq n\text{log } q + 2\lambda$.

Let χ be a noise distribution over $\mathbb{Z}_q$ (it produces small samples, all of magnitude not greater than n). Finally, let $L \in \mathbb{N}$ be an upper bound on the maximal multiplicative depth that the scheme can homomorphically evaluate, say $L \approx \epsilon \log n$.

$\mathsf{KeyGen}(\lambda)$

- pick $\mathbf{s_0}, \ldots, \mathbf{s_L} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$
- pick a matrix $A \overset{\$}{\leftarrow} \mathbb{Z}_q^{m \times n}$
- pick a vector $\mathbf{e} \leftarrow \chi^m$
- compute $\mathbf{b} = A\mathbf{s_0} + 2\mathbf{e}$
- $\mathsf{sk} = \mathbf{s_L}$
- $\mathsf{pk} = (A, \mathbf{b})$

$\mathsf{Encrypt}(\mathsf{pk}, \mu \in \mathcal{M})$

- pick $\mathbf{r} \overset{\$}{\leftarrow} \{0,1\}^m$
- set $\mathbf{v} = A^{\mathrm{T}}\mathbf{r} \in \mathbb{Z}_q^n$
- set $w = \mathbf{b}^{\mathrm{T}}\mathbf{r} + \mu \in \mathbb{Z}_q$
- ciphertext $c = ((\mathbf{v}, w), l)$.

$\mathsf{Decrypt}(\mathsf{sk}, c = ((\mathbf{v}, w), L))$

$$\mu = (w - <\mathbf{v}, \mathbf{s_L}> \mod q) \mod 2$$

Notice that the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_{L-1}$ are used in order to compute the evaluation key, which we omit here. We remark that during the homomorphic evaluation, the scheme generates ciphertexts of the form $c = ((\mathbf{v}, w), l)$, where the tag l indicates the multiplicative level at which the ciphertext has been generated (fresh ciphertexts are tagged with $l = 0$). Note that it always holds that $l \leq L$ due to the bound on the multiplicative depth, and that the output of the homomorphic evaluation of the entire circuit is expected to have $l = L$. As described in [15], the SHE scheme is only required to decrypt ciphertexts that are output by the evaluation step (which we omit here), and those will always have level tag L. Therefore, we always expect a ciphertext of the form $c = ((\mathbf{v}, w), L)$ and decryption is correct.

Apparently, we cannot decrypt level l ciphertexts $c = ((\mathbf{v}, w), l)$, for $1 \leq l < L$, since we are only allowed to decrypt level L ciphertexts. However, we can compute $L - l$ fresh encryptions of 1, namely $c_1, \ldots, c_{L-l}$. Then, we compute $c^* = \mathsf{Evaluate}(\mathsf{pk}, MUL, c, c_1, \ldots, c_{L-l})$ based on the homomorphic property, where MUL is the multiplication circuit. The resulting ciphertext c^* will encrypt the same message as c does, and with a tag level L. In particular, we can decrypt fresh ciphertexts.

5.4.2 Our Key Recovery Attack

We are going to recover the secret key $\mathbf{s_L} \in \mathbb{Z}_q^n$ component-wise, and bit by bit. For ease of notation, we will write $\mathbf{s}$ instead of $\mathbf{s_L}$. More precisely, we write $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$. For every $1 \leq j \leq n$, we have $s_j \in \mathbb{Z}_q$ and therefore s_j can be written with a maximum number N of bits, where $N = \lfloor \log_2(q-1) \rfloor + 1$. We are going to recover the i-th bit of s_j, for all $1 \leq i \leq N$ and for all $1 \leq j \leq n$.

Intuitively, our attack works as follows. We start by finding the first bit of s_j for every $1 \leq j \leq n$; then we will recover the second bit of s_j for every $1 \leq j \leq n$; and we stop until we reach the N-th bit. In order to do so, we have to choose a 'ciphertext' c to be submitted to the decryption oracle. Instead of submitting $c = (\mathbf{v}, w)$ for honestly-generated $\mathbf{v} \in \mathbb{Z}_q^n$ and $w \in \mathbb{Z}_q$, we submit $c^* = (\mathbf{x}, y)$ for some specifically-picked $\mathbf{x} \in \mathbb{Z}_q^n$ and $y \in \mathbb{Z}_q$. We omit to write the level tag since we can always obtain a level tag L from any $l \leq L$.

For any $1 \leq j \leq n$, let $(s_j)_2 := a_{j,N} a_{j,N-1} \cdots a_{j,1}$ be the binary representation of s_j (bits ordered from most significant to least significant). We have $a_{j,i} \in \{0, 1\}$, for all $1 \leq i \leq N$.

Recovering $\mathbf{a_{j,1}}$

We have to choose $\mathbf{x} \in \mathbb{Z}_q^n$ and $y \in \mathbb{Z}_q$ in such a way that $y- < \mathbf{x}, \mathbf{s} > \bmod q = s_j$. To do so, pick $y = 0$ and $\mathbf{x} = (0, \ldots, 0, -1, 0, \ldots, 0)$ (where -1 is in position j). Then, we have $0 - (-1)s_j \bmod \ q = s_j \bmod \ q = s_j$. As a result, by modding out with 2, this will return the last bit $a_{j,1}$ of s_j.

Recovering $\mathbf{a_{j,2}}$

Now that we know the last bit $a_{j,1}$ of s_j, we want to obtain $s_j^{(1)} := (s_j - a_{j,1})/2 \in \mathbb{Z}_q$ whose bit decomposition is the same as the bit decomposition of s_j, but with the last bit removed from it. Then, modding out by 2, we will get the desired bit. This translates to the following condition: find $\mathbf{x} \in \mathbb{Z}_q^n$ and $y \in \mathbb{Z}_q$ such that $y- < \mathbf{x}, \mathbf{s} > \bmod q = (s_j - a_{j,1})/2$. Let $\mathbf{x} = (0, \ldots, 0, x_j, 0, \ldots, 0)$ (with x_j in j-th position). We have to find y and x_j such that $2y - s_j(2x_j + 1) = -a_{j,1} \bmod \ q$. Clearly, the solution is given by $x_j = -2^{-1} \bmod \ q$ and $y = -2^{-1}a_{j,1} \bmod \ q$. By querying the decryption oracle with the 'ciphertext' $c^* := (\mathbf{x}, y)$, we obtain the second-to-last bit $a_{j,2}$ of s_j.

Recovering $\mathbf{a_{j,m}}$, for $\mathbf{1 \leq m \leq N}$

Based on the above two cases, we generalize the procedure. Suppose we have found all bits $a_{j,i}$, for $1 \leq i \leq m-1$. In order to recover the bit $a_{j,m}$, we choose $\mathbf{x} := (0, \ldots, 0, x_j, 0, \ldots, 0) \in \mathbb{Z}_q^n$ and $y \in \mathbb{Z}_q$ as follows: $x_j = -(2^{m-1})^{-1} \bmod \ q$ and $y = -(2^{m-1})^{-1}(\sum_{i=1}^{m-1} 2^{i-1}a_{j,i})$.

5.4.3 Algorithmic Description and Efficiency Analysis

We denote the decryption oracle $\mathcal{O}_D(c) := \mathsf{Decrypt}(\mathsf{sk}, c)$. The ciphertext c is of the form $c = (\mathbf{x}, y)$ (the level tag is omitted), with $\mathbf{x} \in \mathbb{Z}_q^n$, $y \in \mathbb{Z}_q$. For ease of notation, we have also considered the standard vectors $\mathbf{e}_1, \ldots, \mathbf{e}_n \in \mathbb{Z}_q^n$: for every $i = 1, \ldots, n$, $\mathbf{e}_i$ is the $\mathbf{0}$-vector except in position i, where it has value 1, i.e. $\mathbf{e}_i = (e_{i,1}, \ldots, e_{i,n}) = (0, \ldots, 0, 1, 0, \ldots, 0), \quad e_{i,i} = 1, e_{i,j} = 0$ for $j \neq i$. Formally, the attack from Section 5.4.2 can be described by Algorithm 2. It takes as input the integers n, q and returns the secret key vector $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$.

Algorithm 2 Key Recovery Attack

```
input: $n, q \in \mathbb{N}$
$N \leftarrow \lfloor \log_2(q-1) \rfloor + 1$
for $j = 1$ to $n$ do
    for $m = 1$ to $N$ do
        $x_j \leftarrow -(2^{m-1})^{-1} \bmod q$
        $\mathbf{x} \leftarrow x_j \cdot \mathbf{e}_j$
        $y \leftarrow x_j \cdot \sum_{i=1}^{m-1} 2^{i-1} a_{j,i} \bmod q$          $\{\text{if } m = 1,\ y \leftarrow 0\}$
        $a_{j,m} \leftarrow \mathcal{O}_D(\mathbf{x}, y)$
    end for
    $s_j \leftarrow \sum_{m=1}^{N} 2^{m-1} a_{j,m}$
end for
$\mathbf{s} \leftarrow (s_1, \ldots, s_n)$
return $\mathbf{s}$
```

The secret key vector $\mathbf{s} = \mathbf{s_L} \in \mathbb{Z}_q^n$ has n coefficients s_j, and each one of them has length of at most N bits. Now, $n = f(\lambda)$ for a polynomial function $f(\lambda) = \text{poly}(\lambda)$, and $N = \lfloor \log_2(q-1) \rfloor + 1$. We have that $q \in [2^{n^\epsilon}, 2^{n^\epsilon + 1})$, with $\epsilon \in (0,1) \cap \mathbb{R}$ a constant, and

$$\lfloor \log_2(q-1) \rfloor + 1 < \lfloor \log_2 2^{n^\epsilon + 1} \rfloor + 1 = \lfloor n^\epsilon + 1 \rfloor + 1 = \lfloor f(\lambda)^\epsilon + 1 \rfloor + 1 = g(\lambda)$$

where $g(\lambda) = \text{poly}(\lambda)^{\epsilon}$. Therefore, the total number of queries we must perform to recover $\mathbf{s}$ is $n \times N < f(\lambda) \cdot g(\lambda) = \text{poly}(\lambda)^{\epsilon+1}$. Since each query to the decryption oracle reveals one bit of $\mathbf{s}$, our attack is optimal and ends in polynomial time.

5.5 Key Recovery Attack against the BV11a Scheme

In this section, we describe a key recovery attack against the symmetric-key SHE scheme from [14]. The attack also applies to the asymmetric-key SHE scheme.

5.5.1 The BV11a SHE Scheme

Consider primes $q = \text{poly}(\lambda) \in \mathbb{N}$, $t = \text{poly}(\lambda) \in \mathbb{Z}_q^*$. Let $n = \text{poly}(\lambda) \in \mathbb{N}$ and consider a polynomial $f(x) \in \mathbb{Z}[x]$ with $\deg(f) = n + 1$. The message space is $\mathcal{M} = R_t = \mathbb{Z}[x]/(f(x))$. Namely, a message is encoded as a degree n polynomial with coefficients in $\mathbb{Z}_t$. Let χ be an error distribution over the ring $R_q := \mathbb{Z}_q[x]/(f(x))$ and let $D \in \mathbb{N}$, which is related to the maximal degree of homomorphism allowed (and to the maximal ciphertext length). Parameters n, f, q, χ are public.

$\mathsf{Keygen}(\lambda)$

- sample $s \leftarrow \chi$
- $\mathsf{s} = (1, s, s^2, \ldots, s^D) \in R_q^{D+1}$
- $\mathsf{sk} = s$

$\mathsf{Encrypt}(\mathsf{sk}, \mu \in \mathcal{M})$

- sample $a \overset{\$}{\leftarrow} R_q$ and $e \leftarrow \chi$
- compute $(a, b := as + te) \in R_q^2$
- compute $c_0 := b + \mu \in R_q$, $c_1 := -a$
- output $\mathbf{c} = (c_0, c_1) \in R_q^2$

$\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c} = (c_0, \ldots, c_D) \in R_q^{D+1})$

$$\mu = (< \mathbf{c}, \mathbf{s} > \mathrm{mod} q) \bmod t$$

We remark that while the encryption algorithm only generates ciphertexts $\mathbf{c} \in R_q^2$, homomorphic operations (as described in the evaluation algorithm which we omit here) might add more elements to the ciphertext. Thus, the most generic form of a decryptable ciphertext in this scheme is $\mathbf{c} = (c_0, \ldots, c_d) \in R_q^{d+1}$, for $d \leq D$. Notice that 'padding with zeros' does not affect the ciphertext. Namely, $(c_0, \ldots, c_d) \in R_q^{d+1}$ and $(c_0, \ldots, c_d, 0, \ldots, 0) \in R_q^{D+1}$ encrypt the same message $\mu \in R_t$.

5.5.2 Our Key Recovery Attack

We can write $s = s_0 + s_1 x + \cdots + s_n x^n \in \mathbb{Z}_q[x]/(f(x))$ with coefficients $s_j \in \mathbb{Z}_q$, $\forall 0 \leq j \leq n$. We will recover each coefficient s_j separately. Now, each s_j has at most $N := \lfloor \log_2(q-1) \rfloor + 1$ bits; therefore $\#\text{bits}(s) \leq (n+1) \times N = (n+1) \times (\lfloor \log_2(q-1) \rfloor + 1)$ and each query to the oracle decryption will reveal a polynomial $\mu(x) = \mu_0 + \mu_1 x + \cdots + \mu_n x^n \in \mathbb{Z}_t[x]/(f(x))$; we have $\#\text{bits}(\mu) \leq (n+1) \times (\lfloor \log_2(t-1) \rfloor + 1)$. Therefore, the minimum number of oracle queries needed is given by

$$\begin{aligned} & \left\lceil \frac{\#(\text{bits}(s))}{\#(\text{bits revealed by an oracle query})} \right\rceil \\ = & \left\lceil \frac{(n+1) \times (\lfloor \log_2(q-1) \rfloor + 1)}{(n+1) \times (\lfloor \log_2(t-1) \rfloor + 1)} \right\rceil = \left\lceil \frac{\lfloor \log_2(q-1) \rfloor + 1}{\lfloor \log_2(t-1) \rfloor + 1} \right\rceil \end{aligned}$$

We are going to query the decryption oracle with ‘ciphertexts’ of the form $\mathbf{c}_i^* := (h_i, y_i, 0, \ldots, 0) \in R_q^{D+1}$ for some $h_i, y_i \in R_q$. We will describe in detail our attack in the case $t = 2$. An easy generalization for $t \geq 2$ is discussed later.

An easy case: $t = 2$**.**

We expect to query the decryption oracle at least $\lfloor \log_2(q-1) \rfloor + 1$ times and recover s_j, for all $0 \leq j \leq n$, bit by bit. Let $N = \#\text{bits}(s_j) = \lfloor \log_2(q-1) \rfloor + 1$, $\forall 0 \leq j \leq n$; and let $(s_j)_2 = a_{j,N} a_{j,N-1} \cdots a_{j,1}$ be the binary representation of s_j, $\forall 0 \leq j \leq n$ (i.e., $a_{j,i} \in \{0,1\}, \forall 1 \leq i \leq N$ and bits ordered most significant to least significant). For ease of notation, we write $\mathbf{c}^* = (h, y)$ instead of $\mathbf{c}^* = (h, y, 0, \ldots, 0)$.

Recovering $a_{j,1}$**, for all** $0 \leq j \leq n$

For a submitted ’ciphertext’ $\mathbf{c}^* = (h, y)$, decryption works as follows: $< \mathbf{c}^*, \mathbf{s} > \text{mod} 2 = h + ys \bmod 2$. We choose $h = \sum_{j=0}^n 0x^j = 0 \in R_q$ and $y = 1 + \sum_{j=1}^n 0x^j = 1 \in R_q$. The decryption oracle outputs

$$\begin{aligned} s \bmod 2 &= (s_0 \bmod 2) + (s_1 \bmod 2)x + \cdots + (s_n \bmod 2)x^n \\ &= a_{0,1} + a_{1,1}x + \cdots + a_{n,1}x^n \end{aligned}$$

Therefore, we obtain the last bits $a_{j,1}$ for all $1 \leq j \leq n$, which are n bits of s.

Recovering $a_{j,2}$, $\forall 0 \leq j \leq n$

With $a_{j,1}$ for all $0 \leq j \leq n$, we are going to recover $a_{j,2}$, $\forall 0 \leq j \leq n$, as follows. We want to obtain

$$\begin{aligned} s^{(1)} &:= \frac{s - (a_{0,1} + a_{1,1}x + \cdots + a_{n,1}x^n)}{2} \\ &= s_0^{(1)} + s_1^{(1)}x + \cdots + s_n^{(1)}x^n \in \frac{\mathbb{Z}_q[x]}{(f(x))} \end{aligned}$$

for which the bit decomposition of the coefficients $s_j^{(1)}$ is the same as the bit decomposition of s_j, but with the last bit removed from it, for all $0 \leq j \leq n$. Then, by modding out with 2, we will get the desired bits. This translates to the following condition: find $\mathbf{c}^* = (h, y) = (h, y, 0, \ldots, 0) \in R_q^{D+1}$ such that $< \mathbf{c}^*, \mathbf{s} >=$ $s^{(1)} := \frac{s-(a_{0,1}+a_{1,1}x+\cdots+a_{n,1}x^n)}{2}$, from which we obtain $2h + s(2y-1) =$ $-(a_{0,1} + a_{1,1}x + \cdots + a_{n,1}x^n)$. A solution is given by $y = 2^{-1} \in R_q$ and $h = -2^{-1}(a_{0,1} + a_{1,1}x + \cdots + a_{n,1}x^n) \in R_q$. Then, by modding out with 2 the 'decrypted ciphertext' $\mu =< \mathbf{c}^*, \mathbf{s} >$, we recover the second-to-last bits $a_{j,2}$, for all $0 \leq j \leq n$.

Recovering $a_{j,m}$, for $1 \leq m \leq N$, $0 \leq j \leq n$

Suppose we have found all bits $a_{j,i}$, $\forall 1 \leq i \leq m-1$ and $\forall 0 \leq$ $j \leq n$. We want to recover $a_{j,m}$, $\forall 0 \leq j \leq n$. By a recursive argument, we find that we have to submit a 'ciphertext' $\mathbf{c}^* = (h, y)$ such that $y = (2^{m-1})^{-1} \in R_q$ and $h = -(2^{m-1})^{-1} \left(\sum_{j=0}^{n} d_j x^j\right)$ with $d_j = \sum_{i=1}^{m-1} 2^{i-1} a_{j,i}$.

This concludes the attack for the case $t = 2$. Efficiency-wise, the total number of oracle queries is $N = \lfloor \log_2(q-1) \rfloor + 1$, which is optimal.

The general case: $t \geq 2$.

We consider now the general case in which $t \geq 2$ is a prime number in $\mathbb{Z}_q^*$. We want to find $s = s_0 + s_1 x + \cdots + s_n x^n \in \mathbb{Z}_q[x]/(f(x))$ and expect to query the decryption oracle $\left\lceil \frac{\lfloor \log_2(q-1) \rfloor + 1}{\lfloor \log_2(t-1) \rfloor + 1} \right\rceil$ times. With each query to the decryption oracle, we are going to recover $M = \lfloor \log_2(t-1) \rfloor + 1$ bits of s_j, $\forall 0 \leq j \leq n$. The idea is that we are going to recover s_j, for all $0 \leq j \leq n$. In its representation in base t, s_j can be represented with N figures $a_{j,i} \in \{0, 1, \ldots, t-1\}$: $(s_j)_t = a_{j,N} a_{j,N-1} \cdots a_{j,1}$ where $N = \lfloor \log_t(q-1) \rfloor + 1$; each $a_{j,i}$ is bounded by $t-1$, which explains the value $M = \lfloor \log_2(t-1) \rfloor + 1$.

Recovering $a_{j,1}$, $\forall 0 \leq j \leq n$

For a submitted 'ciphertext' $\mathbf{c}^* = (h, y) = (h, y, 0, \ldots, 0) \in R_q^{D+1}$, decryption works as follows: $< \mathbf{c}^*, \mathbf{s} > \bmod t = x + ys \bmod t$. We choose $h = 0 \in R_q$ and $y = 1 \in R_q$. Then, the decryption oracle outputs

$$\begin{aligned} s \bmod t &= (s_0 \bmod t) + (s_1 \bmod t)x + \cdots + (s_n \bmod t)x^n \\ &= a_{0,1} + a_{1,1}x + \cdots + a_{n,1}x^n \end{aligned}$$

as we wanted.

Recovering $a_{j,m}$, $\forall 1 \leq m \leq N$, $\forall 0 \leq j \leq n$

Suppose we know $a_{j,i}$, $\forall 1 \leq i \leq m-1$, $\forall 0 \leq j \leq n$. We want to recover $a_{j,m}$, for all $0 \leq j \leq n$. To do so, we submit to the decryption oracle a 'ciphertext' $\mathbf{c}^* = (h, y)$ such that $y = (t^{m-1})^{-1} \in R_q$, $h = -(t^{m-1})^{-1} \left(\sum_{j=0}^{n} d_j x^j \right)$, $d_j = \sum_{i=1}^{m-1} t^{i-1} a_{j,i}$. It is straightforward to verify that it works and we skip the details here.

5.5.3 Algorithmic Description

Formally, the attack from Section 5.5.2 can be described by Algorithm 3. It takes as input integers $q, t, n, D \in \mathbb{N}$ and a polynomial $f(x) \in \mathbb{Z}[x]$ of degree n. It outputs the secret key vector $\mathbf{s} = (1, s, s^2, \ldots, s^D) \in R_q^{D+1}$, where $R_q := \mathbb{Z}_q[x]/(f(x))$. Let $\mathcal{O}_D(c) := \mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$ be the decryption oracle, where $\mathbf{c} \in R_q^{D+1}$. For given $h, y \in R_q$, let $\mathcal{O}_D(h, y) := \mathcal{O}_D((h, y, 0, \ldots, 0))$.

Algorithm 3 Key Recovery Attack

```
input: $q, t, n, D \in \mathbb{N}$; $f(x) \in \mathbb{Z}[x]$
$N \leftarrow \lfloor \log_t(q-1) \rfloor + 1$
for $m = 1$ to $N$ do
    $y \leftarrow (t^{m-1})^{-1}$ in $R_q$
    $h \leftarrow -y \cdot \sum_{i=1}^{m-1} t^{i-1} r_i$ in $R_q$        {if $m = 1$, $h \leftarrow 0$}
    $r_m \leftarrow \mathcal{O}_D(h, y)$
end for
$s \leftarrow \sum_{i=1}^{N} t^{i-1} r_i$
$\mathbf{s} \leftarrow (1, s, s^2, \ldots, s^D) \in R_q^{D+1}$
return $\mathbf{s}$
```

5.6 Key Recovery Attack against the BGV12 Scheme

The SHE scheme from [12] is closely related to the SHE schemes from [14, 15]. This implies that the attacks from Section 5.5.2 and 5.4.2 can be directly applied against the SHE scheme from [12].

We first remark that the LWE and RLWE problems are syntactically equivalent. They only use different rings ($\mathbb{Z}$ for LWE, and a polynomial ring $\mathbb{Z}[x]/(x^d + 1)$ for RLWE), as well as different vector dimensions over these rings ($n = \text{poly}(\lambda)$ for LWE, $n = 1$ for RLWE). For this reason and to simplify the presentation, the authors of [12] introduced the general learning with errors (GLWE) problem, which is a generalized version of LWE and RLWE.

Definition 13 (GLWE problem)**.** For security parameter λ, let $n = n(\lambda)$ be an integer dimension, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2, let $q = q(\lambda) \geq 2$ be a prime integer, let $R = \mathbb{Z}[x] = (f(x))$ and $R_q = R/qR$, and let $\chi = \chi(\lambda)$ be a distribution over R. The $\mathsf{GLWE}_{n,f,q,\chi}$ problem is to distinguish the following two distributions:

- one samples $(\mathbf{a}_i, b_i)$ uniformly from R_q^{n+1}.
- one first draws $\mathbf{s} \leftarrow R_q^n$ uniformly and then samples $(\mathbf{a}_i, b_i) \in R_q^{n+1}$ by sampling $\mathbf{a}_i \leftarrow R_q^n$ uniformly, $e_i \leftarrow \chi$, and setting $b_i =< \mathbf{a}_i, \mathbf{s} > +e_i$.

The $\mathsf{GLWE}_{n,f,q,\chi}$ assumption is that the $\mathsf{GLWE}_{n,f,q,\chi}$ problem is infeasible.

LWE is GLWE when $d = 1$, and RLWE is GLWE when $n = 1$. Let's review the GLWE-based encryption scheme.

$\mathsf{Setup}(\lambda)$:

- use bit $b \in \{0, 1\}$ to determine whether we are setting parameters for a LWE-based scheme ($d = 1$) or a RLWE-based scheme ($n = 1$).
- choose μ-bit modulus q and d, n, N, χ (all polynomials in λ, μ, b) in order to have a GLWE-based scheme with 2^λ security against known attacks.
- Let $R = \mathbb{Z}[x]/(x^d + 1)$
- Let $\mathsf{params} = (q, d, n, N, \chi)$.

$\mathsf{SecretKeyGen}(\mathsf{params})$**:**

- choose $\mathbf{s}' \leftarrow \chi^n$.
- set $\mathsf{sk} = \mathbf{s} \leftarrow (1, \mathbf{s}'[1], \ldots, \mathbf{s}'[n]) \in R_q^{n+1}$.

$\mathsf{PublicKeyGen}(\mathsf{params}, \mathsf{sk})$:

- input: params and $\mathsf{sk} = \mathbf{s} = (1, \mathbf{s})$ with $\mathbf{s}[0] = 1$ and $\mathbf{s}' \in R_q^n$.
- generate matrix $\mathbf{A}' \overset{\$}{\leftarrow} R_q^{N \times n}$
- generate a vector $\mathbf{e} \leftarrow \chi^N$
- set $b \leftarrow \mathbf{A}'\mathbf{s}' + 2\mathbf{e}$.
- set $\mathbf{A}$ to be the $(n+1)$-column matrix consisting of $\mathbf{b}$ followed by the n columns of $-\mathbf{A}'$. (Remark: $\mathbf{A} \cdot \mathbf{s} = 2\mathbf{e}$.)
- set $\mathsf{pk} = \mathbf{A}$.

$\mathsf{Enc}(\mathsf{params}, \mathsf{pk}, m)$**:** • input: message $m \in R_2$

- set $\mathbf{m} \leftarrow (m, 0, \ldots, 0) \in R_q^{n+1}$
- sample $\mathbf{r} \leftarrow R_2^N$
- output ciphertext $\mathbf{c} \leftarrow \mathbf{m} + \mathbf{A}^T\mathbf{r} \in R_q^{n+1}$.

$\mathsf{Dec}(\mathsf{params}, \mathsf{sk}, c)$**:** Output $m \leftarrow (< \mathbf{c}, \mathbf{s} > \bmod q) \bmod 2$.

The SHE scheme from [12] uses the above GLWE-based encryption scheme as the main building block, and we only need to show attacks against the latter. Depending on which instantiation is chosen (either LWE or RLWE), we can apply one of the key recovery attacks against [15, 14] to the basic GLWE-based encryption scheme.

- If $b = 0$, then $d = 1$, $R = \mathbb{Z}[x]/(x+1) \cong \mathbb{Z}$, and

$$\mathbf{c} := \mathbf{m} + A^{\mathrm{T}}\mathbf{r} = \mathbf{m} + (\mathbf{b} \mid -A'^{\mathrm{T}})\mathbf{r} = \begin{pmatrix} m + \mathbf{b}^{\mathrm{T}}\mathbf{r} \\ -A'^{\mathrm{T}}\mathbf{r} \end{pmatrix}$$

  which can be written as $\mathbf{c} = \binom{w}{-\mathbf{v}} \in \mathbb{Z}_q^{n+1}$. For decryption we have

$$\begin{aligned} m &:= (<\mathbf{c}, \mathbf{s}> \bmod q) \bmod 2 \\ &= (m + \mathbf{b}^{\mathrm{T}}\mathbf{r} + <-A'^{\mathrm{T}}\mathbf{r}, \mathbf{s}> \bmod q) \bmod 2 \end{aligned}$$

  which is $(w - <\mathbf{v}, \mathbf{s}_L> \bmod q) \bmod 2$. The secret key can be recovered by directly applying the key recovery attack from Section 5.4.2.

- If $b = 1$, then $n = 1$, $R = \mathbb{Z}[x]/(x^d + 1)$. The scheme is slightly different from the BV11a SHE scheme just for the encryption part, but the setup, the key generation and the decryption steps are the same. Therefore, our key recovery attack can be applied. Precisely, to recover the secret polynomial $\mathbf{s} := s(x) = s_0 + s_1x + \cdots + s_{d-1}x^{d-1} \in \mathbb{Z}[x]/(x^d + 1)$, one could directly use our key recovery attack from Section 5.5.2 with the following settings: $D \leftarrow 1, n \leftarrow d-1, t \leftarrow 2$.

5.7 Key Recovery Attack against the Bra12 SHE Scheme

In this section, we describe a key recovery attack against the SHE scheme from [11]. The scheme uses, as a building block, Regev's [68] public-key encryption scheme. It is then enough to show a key recovery attack on

Regev's scheme since the full [11] can be attacked exactly as the basic Regev's encryption scheme (the only differences between the two schemes are in the evaluation step which is missing in Regev's scheme).

Let's first recall Regev's encryption scheme. In this scheme, let $n := \lambda$ be the security parameter.

5.7.1 The Bra12 SHE Scheme (Regev's Encryption Scheme)

Let q be a prime number and let $\chi = \chi(n)$ be a distribution ensemble over $\mathbb{Z}$. The message space is $\mathcal{M} = \{0, 1\}$. As claimed in [68], choosing q such that $n^2 \leq q \leq 2n^2$ is enough for security (in particular, $q = \mathrm{poly}(n)$; for other parameters settings, see [68]).

$\mathsf{SecretKeyGen}(n)$:

- Sample $\mathbf{s} := (s_1, \ldots, s_n) \xleftarrow{\$} \mathbb{Z}_q^n$
- Output $\mathsf{sk} = \mathbf{s}$

$\mathsf{PublicKeyGen}(\mathbf{s})$:

- Let $N := (n + 1) \cdot (\log q + O(1))$
- Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{N \times n}$
- Sample $\mathbf{e} \leftarrow \chi^N$
- Compute $\mathbf{b} := \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \bmod q$
- Define $\mathbf{P} := [\mathbf{b} | -\mathbf{A}] \in \mathbb{Z}_q^{N \times (n+1)}$
- Output $\mathsf{pk} = \mathbf{P}$

$\mathsf{Encrypt}(\mathsf{pk}, m \in \{0, 1\})$:

- Sample $\mathbf{r} \in \{0, 1\}^N$
- Let $\mathbf{m} := (m, 0, \ldots, 0) \in \{0, 1\}^{n+1}$

$$\mathbf{c} := \mathbf{P}^T \cdot \mathbf{r} + \lfloor q/2 \rfloor \cdot \mathbf{m} \in \mathbb{Z}_q^{n+1}$$

$\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$:

$$m := \left\lfloor \frac{2}{q} \cdot (< \mathbf{c}, (1, \mathbf{s}) > \bmod q) \right\rceil \bmod 2$$

5.7.2 Our Key Recovery Attack

Recall that we defined the rounding function $\lfloor \cdot \rceil$ such that $\lfloor m + 1/2 \rceil := m + 1$ for every $m \in \mathbb{N}$. The following attack works also, with trivial modifications, in case we define $\lfloor m + 1/2 \rceil := m$. In this section, for a given ciphertext $\mathbf{c}$ we use notation $D(\mathbf{c})$ instead of $\mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$.

We will start by describing how to recover s_1. An easy generalization will allow to recover s_j, $\forall j = 1, 2, \ldots, n$. We are going to submit to the decryption oracle 'ciphertexts' of the form $\mathbf{c} = (c_1, c_2, \ldots, c_{n+1}) \in \mathbb{Z}_q^{n+1}$. It holds

$$< \mathbf{c}, (1, \mathbf{s}) > \bmod q = c_1 + c_2 s_1 + c_3 s_2 + \cdots + c_{n+1} s_n \bmod q$$

Choose $\mathbf{c} = (0, 1, 0, \ldots, 0)$, i.e. $c_2 = 1$ and $c_i = 0$, for $i = 1, \ldots, n+1$ and $i \neq 2$. Then $< \mathbf{c}, (1, \mathbf{s}) > \bmod q = s_1 \bmod q = s_1$. (Recall that $s_j \leq q - 1$, $\forall j = 1, 2, \ldots, n$.) Then

$$D(\mathbf{c}) = \left\lfloor \frac{2}{q} s_1 \right\rceil \bmod 2$$

Now, since $0 \leq s_1 < q$, we have $0 \leq \frac{2}{q} s_1 < \frac{2}{q} q = 2$. Let $u = \left\lfloor \frac{2}{q} s_1 \right\rceil$; then we have $u \in \{0, 1, 2\}$. In particular, it is easy to see that

- $u = 0 \Leftrightarrow 0 \leq s_1 < \frac{q}{4}$
- $u = 1 \Leftrightarrow \frac{q}{4} \leq s_1 < \frac{3q}{4}$
- $u = 2 \Leftrightarrow \frac{3q}{4} \leq s_1 \leq q - 1$

Remember that q is prime, so in particular $\frac{q}{4}, \frac{3q}{4} \notin \mathbb{N}$. Since $D(\mathbf{c}) = u \bmod 2$, we have

- $D(\mathbf{c}) = 1 \Leftrightarrow \frac{q}{4} < s_1 < \frac{3q}{4}$
- $D(\mathbf{c}) = 0 \Leftrightarrow 0 \leq s_1 < \frac{q}{4}$ or $\frac{3q}{4} < s_1 \leq q-1$

Having these considerations in mind, we recover s_1 like follows.

Recovering s_1.

Select $\mathbf{c} = (0, 1, 0, \ldots, 0) \in \mathbb{Z}_q^{n+1}$, i.e. $c_2 = 1$ and $c_i = 0$, $\forall i = 1, \ldots, n+1$, $i \neq 2$. Submit $\mathbf{c}$ to the decryption oracle. We have two case to consider.

Case 1: $D(\mathbf{c}) = 1$. Then we know that

$$\frac{q}{4} < s_1 < \frac{3q}{4} \tag{5.1}$$

Now select $\mathbf{c} = (1, 1, 0, \ldots, 0) \in \mathbb{Z}_q^{n+1}$. Then $< \mathbf{c}, (1, \mathbf{s}) >= 1 + s_1 \bmod q$. There are two cases to consider:

1.1 if $D(\mathbf{c}) = 0$, then it must be

$$\frac{3q}{4} < 1 + s_1 \tag{5.2}$$

Conditions 5.1 and 5.2 together imply that s_1 is the biggest integer smaller than $\frac{3q}{4}$, i.e. $s_1 = \lfloor \frac{3q}{4} \rfloor$.

1.2 if $D(\mathbf{c}) = 1$, then we still have

$$\frac{q}{4} < 1 + s_1 < \frac{3q}{4} \tag{5.3}$$

We then select $\mathbf{c} = (2, 1, 0, \ldots, 0) \in \mathbb{Z}_q^{n+1}$ and submit it to the decryption oracle. Similarly as above, we have $< \mathbf{c}, (1, \mathbf{s}) >= 2 + s_1 \bmod q$. Again, there are two cases to consider:

1.2.1 if $D(\mathbf{c}) = 0$, then it must be

$$\frac{3q}{4} < 2 + s_1 \tag{5.4}$$

Conditions 5.3 and 5.4 together imply that $1 + s_1 = \lfloor \frac{3q}{4} \rfloor$, i.e. $s_1 = \lfloor \frac{3q}{4} \rfloor - 1$.

1.2.2 if $D(\mathbf{c}) = 1$, then we still have

$$\frac{q}{4} < 2 + s_1 < \frac{3q}{4}$$

We keep reasoning this way, submitting to the decryption oracle 'ciphertexts' $\mathbf{c}_i = (c_{1,i}, 1, 0, \ldots, 0) \in \mathbb{Z}_q^{n+1}$, for increasing values $c_{1,i} = 1, 2, 3, \ldots$ until we obtain $D(\mathbf{c}_i) = 0$. Then we will have

$$s_1 = \lfloor \frac{3q}{4} \rfloor - c_{1,i} + 1$$

We notice that, in the worst case, i.e. when $s_1 = \lceil \frac{q}{4} \rceil$, we have to query the decryption oracle at most $M_1 := \lceil \frac{3q}{4} \rceil - \lceil \frac{q}{4} \rceil$ times. Therefore, in the worst case the total number of oracle queries is

$$T_1 := 1 + M_1 = 1 + \lceil \frac{3q}{4} \rceil - \lceil \frac{q}{4} \rceil \approx \frac{q}{2}$$

Case 2: $D(\mathbf{c}) = 0$. Then we know that s_1 is such that

(2.1) $0 \leq s_1 < \frac{q}{4}$ or

(2.2) $\frac{3q}{4} < s_1 \leq q - 1$

We use techniques as before, but we have to be more careful since now we have to understand in which case we are among (2.1) or (2.2).

As before, we select $\mathbf{c} = (1, 1, 0, \ldots, 0) \in \mathbb{Z}_q^{n+1}$. Then $< \mathbf{c}, (1, \mathbf{s}) >= 1 + s_1 \bmod q$.

The idea is similar to case 1: we keep submitting to the decryption oracle 'ciphertexts' $\mathbf{c}_i = (c_{1,i}, 1, 0, \ldots, 0) \in \mathbb{Z}_q^{n+1}$, for increasing values $c_{1,i} = 1, 2, 3, \ldots$, until $D(\mathbf{c}_i) = 1$. When we will receive $D(\mathbf{c}_i) = 1$, we will know that $s_1 + c_{1,i} > \frac{q}{4}$. The exact value $c_{1,i}$ will tell us in which of the cases (2.1) or (2.2) we were at the beginning. In fact,

- in case (2.1) we will get $D(\mathbf{c}_i) = 1$ after a number of oracle queries M_2' such that

$$1 \leq M_2' \leq \left\lceil \frac{q}{4} \right\rceil$$

  where $M_2' = 1$ when $s_1 = \left\lfloor \frac{q}{4} \right\rfloor$ and $M_2' = \left\lceil \frac{q}{4} \right\rceil$ when $s_1 = 0$.

- in case (2.2) the number M_2'' of oracle queries needed in order to obtain $D(\mathbf{c}_i) = 1$ is such that

$$1 + \left\lceil \frac{q}{4} \right\rceil \leq M_2'' \leq q - \left\lceil \frac{3q}{4} \right\rceil + \left\lceil \frac{q}{4} \right\rceil$$

  where $M_2'' = 1 + \left\lceil \frac{q}{4} \right\rceil$ when $s_1 = q - 1$ and $M_2'' = q - \left\lceil \frac{3q}{4} \right\rceil + \left\lceil \frac{q}{4} \right\rceil$ when $s_1 = \left\lceil \frac{3q}{4} \right\rceil$.

Therefore, consider the first value $c_{1,i}$ such that $D(\mathbf{c}_i) = 1$.

- if $1 \leq c_{1,i} \leq \left\lceil \frac{q}{4} \right\rceil$, we are in case (2.1) and

$$s_1 = \left\lfloor \frac{q}{4} \right\rfloor - c_{1,i} + 1$$

- if $1 + \left\lceil \frac{q}{4} \right\rceil \leq c_{1,i} \leq q - \left\lceil \frac{3q}{4} \right\rceil + \left\lceil \frac{q}{4} \right\rceil$ we are in case (2.2) and

$$s_1 = q - c_{1,i} + \left\lceil \frac{q}{4} \right\rceil$$

We notice that, in the worst case (i.e., when $s_1 = \left\lceil \frac{3q}{4} \right\rceil$) we need to query the decryption oracle $M_0 := q - \left\lceil \frac{3q}{4} \right\rceil + \left\lceil \frac{q}{4} \right\rceil$ times. (Notice that $M_0 = q - M_1$.) Therefore, in case 2, in the worst case the total number of oracle queries is

$$T_2 := 1 + M_0 = 1 + q - \left\lceil \frac{3q}{4} \right\rceil + \left\lceil \frac{q}{4} \right\rceil \approx \frac{q}{2}$$

So in both cases 1 and 2, the total number of oracle queries needed to recover s_1 is $\approx \frac{q}{2}$.

Remark 4. We can provide an exact simpler formula for T_1 and T_2. Recall that $q \geq 2$ is prime; we can reasonably assume q odd. Then one can check that

- if $q \equiv 1 \bmod 4$

$$M_1 = \frac{q-1}{2}, M_0 = \frac{q+1}{2}, T_1 = \frac{q+1}{2}, T_2 = \frac{q+3}{2}$$

- if $q \equiv 3 \bmod 4$

$$M_1 = \frac{q+1}{2}, M_0 = \frac{q-1}{2}, T_1 = \frac{q+3}{2}, T_2 = \frac{q+1}{2}$$

In particular, the total number T_{tot} of oracle queries needed to recover s_1 is $T_{\text{tot}} \leq \frac{q+3}{2}$.

An optimization. We could optimize the previous algorithm like follows. Let $b := D(\mathbf{c}) \in \{0, 1\}$, where $\mathbf{c} = (0, 1, 0, \dots, 0) \in \mathbb{Z}_q^{n+1}$. Our previous strategy was to submit 'ciphertexts' $\mathbf{c}_i := (c_{1,i}, 1, 0, \dots, 0) \in \mathbb{Z}_q^{n+1}$ for *increasing* values $c_{1,i} = i$, for $i = 1, 2, \dots, M_b$.

We modify our strategy and choose the first value $c_{1,1}$ in the middle of the interval $[1, M_b]$. Then, if $D(\mathbf{c}_1) = 1 + b \bmod 2$ we choose $c_{1,2}$ in the middle of the interval $[1, c_{1,1}]$; otherwise, if $D(\mathbf{c}_1) = b \bmod 2$, we choose $c_{1,2}$ in the middle of the interval $[c_{1,1} + 1, M_b]$. Keep reasoning this way, we will obtain s_1 in $\lfloor \log_2(M_b) + 1 \rfloor \approx \log_2(q/2)$ oracle queries.

Recovering s_j, for $j = 1, \dots, n$.
Similarly, and more in general, we can recover s_j, for $j = 1, 2, \dots, n$. In this case, the 'ciphertext' to submit is $\mathbf{c} = (c_1, c_2, \dots, c_{n+1}) \in \mathbb{Z}_q^{n+1}$ with

$$c_k = \begin{cases} 0 & \text{if } k = 1 \text{ and it is the first query to the decryption oracle} \\ c_{1,i} & \text{if } k = 1 \text{ and it is not the first query, } 1 \leq c_{1,i} \leq M_b \\ 1 & \text{if } k = j + 1 \\ 0 & \text{if } k \notin \{1, j + 1\} \end{cases}$$

5.7.3 Algorithmic Description and Efficiency

For a given vector $\mathbf{c} = (c_1, \dots, c_{n+1}) \in \mathbb{Z}_q^{n+1}$, we denote the decryption oracle as $\mathcal{O}_D(\mathbf{c}) := \mathsf{Decrypt}(\mathsf{sk}, \mathbf{c})$. For ease of notation, we define the following function. Let $j \in \{1, 2, \dots, n\}$, and $a, b \in \mathbb{Z}_q$; define the function $f_j : \mathbb{Z}_q^2 \to \mathbb{Z}_q^{n+1}$ such that $f_j(b_1, b_2) = (a_1, \dots, a_{n+1})$ with $a_1 = b_1$, $a_{j+1} = b_2$ and $a_k = 0$ for $k \neq 1, j+1$. Algorithm 4 takes as input the integers n, q and returns the secret key $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$. Algorithm 5 is the optimized version of it.

Algorithm 4 Key-Recovery Attack

```
input: $q, n \in \mathbb{N}$
for $j = 1$ to $n$ do
    $\mathbf{c} \leftarrow f_j(0,1)$, $b \leftarrow \mathcal{O}_D(\mathbf{c})$, $b' \leftarrow b$, $i \leftarrow 0$
    while $b' = b$ do
        $i = i + 1$
        $\mathbf{c} \leftarrow f_j(i,1)$
        $b' \leftarrow \mathcal{O}_D(\mathbf{c})$
    end while
    if $b = 1$ then
        $s_j \leftarrow \left\lfloor \frac{3q}{4} \right\rfloor - i + 1$
    else if $b = 0$ then
        if $1 \leq i \leq \lceil \frac{q}{4} \rceil$ then
            $s_j \leftarrow \lfloor \frac{q}{4} \rfloor - i + 1$
        else if $1 + \lceil \frac{q}{4} \rceil \leq i \leq q - \left\lceil \frac{3q}{4} \right\rceil + \lceil \frac{q}{4} \rceil$ then
            $s_j \leftarrow q - i + \lceil \frac{q}{4} \rceil$
        end if
    end if
end for
return $\mathbf{s} := (s_1, \ldots, s_n)$
```

Notice that $\max(M_0, M_1) = (q+1)/2 =: M$. Therefore, in the worst case the total number T_{tot} of oracle queries needed to recover $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_q^n$ is

$$T_{\text{tot}} \leq n \cdot (1 + M) = n \cdot \frac{q+3}{2} \approx n \cdot \frac{q}{2}$$

In the optimized version, the total number $T_{\text{tot}}^{\text{opt}}$ of oracle queries is

$$T_{\text{tot}}^{\text{opt}} \leq n \cdot (1 + \lfloor \log_2(M) + 1 \rfloor) \approx n \cdot (1 + \log_2(q/2))$$

Therefore, our optimized key recovery algorithm is indeed optimal since the number of oracle queries needed to recover the secret key is not greater than the bits of the secret key (and one oracle query reveals on bit at a

Algorithm 5 Optimized Key-Recovery Attack

```
input: $q, n \in \mathbb{N}$
if $q \equiv 1 \bmod 4$ then
    $M_1 \leftarrow \frac{q-1}{2}$, $M_0 \leftarrow \frac{q+1}{2}$
else if $q \equiv 3 \bmod 4$ then
    $M_1 \leftarrow \frac{q+1}{2}$, $M_0 \leftarrow \frac{q-1}{2}$
end if
for $j = 1$ to $n$ do
    $\mathbf{c} \leftarrow f_j(0,1)$, $b \leftarrow \mathcal{O}_D(\mathbf{c})$, $L \leftarrow 1$, $U \leftarrow M_b$
    while $L \neq U$ do
        $i \leftarrow \left\lfloor \frac{L+U}{2} \right\rfloor$
        $\mathbf{c} \leftarrow f_j(i,1)$
        $b' \leftarrow \mathcal{O}_D(\mathbf{c})$
        if $b' \neq b$ then
            $U \leftarrow i$
        else if $b' = b$ then
            $L \leftarrow i+1$
        end if
    end while
    if $b = 1$ then
        $s_j \leftarrow \left\lfloor \frac{3q}{4} \right\rfloor - L + 1$
    else if $b = 0$ then
        if $1 \leq L \leq \left\lceil \frac{q}{4} \right\rceil$ then
            $s_j \leftarrow \left\lfloor \frac{q}{4} \right\rfloor - L + 1$
        else if $1 + \left\lceil \frac{q}{4} \right\rceil \leq L \leq q - \left\lceil \frac{3q}{4} \right\rceil + \left\lceil \frac{q}{4} \right\rceil$ then
            $s_j \leftarrow q - L + \left\lceil \frac{q}{4} \right\rceil$
        end if
    end if
end for

return  $\mathbf{s} := (s_1, \ldots, s_n)$
```

time). In fact:

$$\#\text{bits in } \mathsf{sk} = n \cdot (1 + \lfloor \log_2(q-1) \rfloor)$$

$$\begin{aligned} T_{\text{tot}}^{\text{opt}} \leq n \cdot \left(1 + \left\lfloor \log_2\left(\frac{q+1}{2}\right) + 1 \right\rfloor\right) &= \left(1 + \left\lfloor \log_2\left(\frac{q+1}{2} \cdot 2\right) \right\rfloor\right) \\ &= n \cdot (1 + \lfloor \log_2(q+1) \rfloor) \end{aligned}$$

In particular the above algorithm is polynomial in the security parameter n: recall that the suggested parameters in Regev's encryption scheme [68] for q are $n^2 \leq q \leq 2n^2$; therefore we have $T_{\text{tot}}^{\text{opt}} = O(n\log\, n)$.

5.8 Key Recovery Attack against the GSW13 SHE Scheme

In this section, we describe a key recovery attack against the SHE scheme from [44]. We first give some useful preliminary definitions. Let $q, k \in \mathbb{N}$. Let l be the bit-length of q, i.e. $l = \lfloor \log_2 q \rfloor + 1$, and let $N = k \cdot l$. Consider a vector $\mathbf{a} := (a_1, \ldots, a_k) \in \mathbb{Z}_q^k$, and let $(a_i)_2 := a_{i,0}a_{i,1}\ldots a_{i,l-1}$ be the binary decomposition of a_i (bit ordered least to most significant), for every $i = 1, \ldots, k$. We define

$$\mathsf{BitDecomp}(\mathbf{a}) := (a_{1,0}, \ldots, a_{1,l-1}, \ldots, a_{k,0}, \ldots, a_{k,l-1}) \in \mathbb{Z}_q^N$$

For a given $\mathbf{a}' := (a_{1,0}, \ldots, a_{1,l-1}, \ldots, a_{k,0}, \ldots, a_{k,l-1}) \in \mathbb{Z}_q^N$, let

$$\mathsf{BitDecomp}^{-1}(\mathbf{a}') := (\sum_{j=0}^{l-1} 2^j \cdot a_{1,j}, \ldots, \sum_{j=0}^{l-1} 2^j \cdot a_{k,j}) \in \mathbb{Z}_q^k$$

We notice explicitly that $\mathbf{a}'$ does not necessarily lie in $\{0,1\}^N$, but when it does then $\mathsf{BitDecomp}^{-1}$ is the inverse of $\mathsf{BitDecomp}$. For $\mathbf{a}' \in \mathbb{Z}_q^N$, we define

$$\mathsf{Flatten}(\mathbf{a}') := \mathsf{BitDecomp}(\mathsf{BitDecomp}^{-1}(\mathbf{a}')) \in \mathbb{Z}_q^N$$

When A is a matrix, let $\mathsf{BitDecomp}(A), \mathsf{BitDecomp}^{-1}(A), \mathsf{Flatten}(A)$ be the matrix formed by applying the operation to each row of A separately. Finally, for $\mathbf{b} := (b_1, \ldots, b_k) \in \mathbb{Z}_q$ let

$$\mathsf{PowersOf2}(\mathbf{b}) := (b_1, 2b_1, \ldots, 2^{l-1}b_1, \ldots, b_k, 2b_k, \ldots, 2^{l-1}b_k) \in \mathbb{Z}_q^N$$

It is easy to see that, for $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^k$ and for $\mathbf{a}' \in \mathbb{Z}_q^N$,

$$\begin{aligned} < \mathsf{BitDecomp}(\mathbf{a}), \mathsf{Powersof2}(\mathbf{b}) > &= < \mathbf{a}, \mathbf{b} > \\ < \mathbf{a}', \mathsf{Powersof2}(\mathbf{b}) > &= < \mathsf{BitDecomp}^{-1}(\mathbf{a}'), \mathbf{b} > \\ &= < \mathsf{Flatten}(\mathbf{a}'), \mathsf{Powersof2}(\mathbf{b}) > \end{aligned}$$

5.8.1 The GSW13 SHE Scheme

The message space is $\mathcal{M} = \mathbb{Z}_q$ for a given modulus q with # bits$(q) = \kappa = \kappa(\lambda, L)$. Let $n = n(\lambda)$ be the lattice dimension and let $\chi = \chi(\lambda)$ be the error distribution over $\mathbb{Z}_q$ (chosen appropriately for LWE: it must achieve at least 2^λ security against known attacks). Choose $m = m(\lambda) = O(n\log q)$. So the parameters used in all algorithms are n, q, χ, m. We have that $l = \lfloor \log q \rfloor + 1$ is the number of bits of q, and we let $N = (n+1) \cdot l$.

$\mathsf{Keygen}(\lambda)$:

- sample $\mathbf{t} := (t_1, \ldots, t_n) \leftarrow \mathbb{Z}_q^n$
- $\mathsf{sk} := \mathbf{s} \leftarrow (1, -t_1, \ldots, -t_n) \in \mathbb{Z}_q^{n+1}$

- let $\mathbf{v} = \mathsf{Powersof2}(\mathbf{s}) \in \mathbb{Z}_q^N$; see [1]
- sample a matrix $B \overset{\$}{\leftarrow} \mathbb{Z}_q^{m \times n}$
- sample a vector $\mathbf{e} \leftarrow \chi$, $\mathbf{e} \in \mathbb{Z}_q^m$
- set $\mathbf{b} := B \cdot \mathbf{t} + \mathbf{e} =: (b_1, \ldots, b_m) \in \mathbb{Z}_q^m$.
- set A to be the $(n+1)$-column matrix consisting of $\mathbf{b}$ followed by the n columns of B

$$A = (\mathbf{b} \mid B) \in \mathbb{Z}_q^{m \times (n+1)}$$

- $\mathsf{pk} := A$.

We remark that $A \cdot \mathbf{s} = \mathbf{e}$.

$\mathsf{Encrypt}(\mathsf{pk}, \mu \in \mathcal{M})$**:**

- sample a matrix $R \overset{\$}{\leftarrow} \{0,1\}^{N \times m}$
- output the ciphertext

$$C = \mathsf{Flatten}(\mu \cdot I_N + \mathsf{BitDecomp}(R \cdot A)) \in \mathbb{Z}_q^{N \times N}$$

$\mathsf{Decrypt}(\mathsf{sk}, C)$**:**

- observe that the first l coefficients of $\mathbf{v}$ are $1, 2, \ldots, 2^{l-2}$
- among these coefficients, let $v_i = 2^i$ be in $(q/4, q/2]$
- let C_i be the i-th row of C
- compute $x_i :=< C_i, \mathbf{v} >$

[1] $\mathbf{v} = \mathsf{Powersof2}(\mathbf{s}) = (s_1, 2s_1, \ldots, 2^{l-1}s_1, s_2, \ldots, 2^{l-1}s_2, \ldots, s_{n+1}, 2s_{n+1}, \ldots, 2^{l-1}s_{n+1}) = (1, 2, \ldots, 2^{l-1}, -t_1, -2t_1, \ldots, -2^{l-1}t_1, \ldots, -t_n, -2t_n, \ldots, -2^{l-1}t_n) \in \mathbb{Z}_q^{(n+1)l} = \mathbb{Z}_q^N$

- output $\mu' := \lfloor x_i / v_i \rceil$

The Decrypt algorithm can recover the message μ when it is in a 'small space' ($q = 2$, i.e. $\mathcal{M} = \mathbb{Z}_2$). For an algorithm that can recover any $\mu \in \mathbb{Z}_q$, we refer to the MPDec algorithm as described (as a special case) in [44] and in [59]. If the ciphertext is generated correctly, it is not difficult to show that $C \cdot \mathbf{v} = \mu \cdot \mathbf{v} + R \cdot A \cdot \mathbf{s} = \mu \cdot \mathbf{v} + R \cdot \mathbf{e} \in \mathbb{Z}_q^N$.

Now, the Decrypt algorithm uses only the i-th coefficient of the vector $C \cdot \mathbf{v} \in \mathbb{Z}_q^N$, i.e. $< C_i, \mathbf{v} >= \mu \cdot v_i + < R_i, \mathbf{e} > \in \mathbb{Z}_q$. Moreover, in the Decrypt step, i has to be such that $v_i := 2^i \in (q/4, q/2]$, with $i \in [1, 2, \ldots, 2^{l-1}]$. Now remember that $l = \lfloor \log q \rfloor + 1$ equals the number of bits of q. Hence we have

$$2^{l-3} \leq \frac{q}{4} < 2^{l-2} \leq \frac{q}{2} < 2^{l-1} \leq q < 2^l$$

Therefore the only possible value for $2^i \in (q/4, q/2]$ is 2^{l-2}. For this reason, Decrypt can be simply rewritten as

Decrypt(sk, C):

- let C_{l-2} be the $(l-2)$-th row of C
- compute $x_{l-2} :=< C_{l-2}, \mathbf{v} >$
- output $\mu' := \lfloor x_{l-2} / 2^{l-2} \rceil$

One could think of outputting as ciphertext only the $(l-2)$-th row C_{l-2} of the matrix C; this is actually not possible since the full matrix is still needed in order to perform the homomorphic operations (in particular, the multiplication of two ciphertexts). We will not discuss them here; see [44].

5.8.2 Our Key Recovery Attack

We are going to recover bit by bit each coefficient t_i of the secret vector $\mathbf{t} := (t_1, \ldots, t_n) \in \mathbb{Z}_q^n$. For every $1 \leq i \leq n$, let $\mathsf{BitDecomp}(t_i) := (t_{i,0}, t_{i,1}, \ldots, t_{i,l-1}) \in \mathbb{Z}_q^l$ bits ordered from least to most significant. We explicitly remark that $t_i = \sum_{j=0}^{l-1} 2^j t_{i,j}$. We will proceed as follows: start with $i = 1$ and recover, in this order, the bits from most to least significant. Then continue with $i = 2$, and so on until $i = n$. Let $x \in \mathbb{Z}_q$. Since $\#\text{bits}(q) = l$, we have $x \leq q - 1 \leq 2^l - 2$. Moreover, we have $\#\text{bits}(x) \leq \lfloor \log_2(q-1) \rfloor + 1 := l^*$. We have $l^* = l$ if q is not a power of 2, i.e. if $q \neq 2^h$, for any $h \in \{1, 2, \ldots, l-1\}$. Otherwise, $l^* = l - 1$. We will not distinguish between these two cases: just remark that if $l^* = l - 1$, then $t_{i,l-1} = 0$ for all $i \in \{1, 2, \ldots, n\}$.

Recovering $\mathsf{BitDecomp}(\mathsf{t}_1)$

We start by recovering $\mathsf{BitDecomp}(\mathsf{t}_1)$. The trickiest part is to recover the most significant bit. We start by recovering $t_{1,l-1}, t_{1,l-2}, t_{1,l-3}$. We have to choose, and submit to the decryption oracle, a matrix $C \in \mathbb{Z}_q^{N \times N}$. Then the oracle will compute $x =< C_{l-2}, \mathbf{v} >$ and will output the rounded value $\mu = \lfloor x/2^{l-2} \rceil$. Our attack works also, with a trivial modification, in the case we define the rounding function such that $\lfloor n + 1/2 \rceil := n$, for every $n \in \mathbb{N}$. Our strategy is to submit a matrix C whose entries are all 0 except for the $(l-2)$-th row C_{l-2}. Let $\mathbf{y} = (y_1, \ldots, y_N) \in \mathbb{Z}_q^N$ be the vector representing C_{l-2}.

We select $\mathbf{y} = (0, \ldots, 0, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$ where -1 is in $l + 1$-th position, i.e.

$$y_i = \begin{cases} -1 & \text{if } i = l + 1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= -v_{l+1} = t_1 \in \mathbb{Z}_q$ and $\mu = \lfloor t_1/2^{l-2} \rceil$. There are two cases.

1. $\mu = 0$. In this case, we have $0 \leq \frac{t_1}{2^{l-2}} < \frac{1}{2}$ i.e. $t_1 < 2^{l-3} = \sum_{j=0}^{l-4} 2^j + 1$. Then it must be $\boxed{t_{1,l-1} = t_{1,l-2} = t_{1,l-3} = 0}$.

2. $1 \leq \mu \leq 4$. In particular, $2^{l-3} \leq t_1 \leq 2^l - 2$. Then we have

$$(t_{1,l-1}, t_{1,l-2}, t_{1,l-3}) \in \{0,1\}^3 \backslash \{(0,0,0)\} \tag{5.5}$$

Next, query the decryption oracle with

$$\mathbf{y} = (0, \ldots, 0, -1, 0, 0, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

with -1 in $(l-2)$-th and $(l+1)$-th positions:

$$y_i = \begin{cases} -1 & \text{if } i = l-2 \text{ or } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= t_1 - 2^{l-3} \geq 0$ and $\mu = \left\lfloor \frac{t_1 - 2^{l-3}}{2^{l-2}} \right\rceil$. There are two cases:

2.1. $\mu = 0$. In this case, we have $0 \leq \frac{t_1 - 2^{l-3}}{2^{l-2}} < \frac{1}{2}$ i.e. $2^{l-3} \leq t_1 < 2^{l-2} = \sum_{j=0}^{l-3} 2^j + 1$. Then it must be $\boxed{t_{1,l-1} = t_{1,l-2} = 0}$. Condition (5.5) implies that $\boxed{t_{1,l-3} = 1}$.

2.2. $1 \leq \mu \leq 3$. In particular, $2^{l-2} \leq t_1 \leq 2^l - 2$. Then we have

$$(t_{1,l-1}, t_{1,l-2}) \in \{0,1\}^2 \backslash \{(0,0)\} \tag{5.6}$$

Next, query the decryption oracle with

$$\mathbf{y} = (0, \dots, 0, -1, 0, -1, 0, \dots, 0) \in \mathbb{Z}_q^N$$

with -1 in $(l-1)$-th and $(l+1)$-th positions:

$$y_i = \begin{cases} -1 & \text{if } i = l-1 \text{ or } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= t_1 - 2^{l-2} \geq 0$ and $\mu = \left\lfloor \frac{t_1 - 2^{l-2}}{2^{l-2}} \right\rceil$. There are two cases:

2.2.1. $\mu = 0$. In this case, we have $0 \leq \frac{t_1 - 2^{l-2}}{2^{l-2}} < \frac{1}{2}$ and $2^{l-2} \leq t_1 < 2^{l-2} + 2^{l-3} < 2^{l-1}$. This means that $\boxed{t_{1,l-1} = 0}$. Therefore, condition (5.6) implies that $\boxed{t_{1,l-2} = 1}$. Moreover, since we have $0 \leq t_1 - 2^{l-2} < 2^{l-3}$, we have that $\boxed{t_{1,l-3} = 0}$.

2.2.2. $1 \leq \mu \leq 2$. In particular, $2^{l-3} + 2^{l-2} \leq t_1$.

Next, query the decryption oracle with

$$\mathbf{y} = (0, \dots, 0, -1, -1, 0, -1, 0, \dots, 0) \in \mathbb{Z}_q^N$$

with -1 in $(l-2)$-th, $(l-1)$-th and $(l+1)$-th positions:

$$y_i = \begin{cases} -1 & \text{if } i = l-2, i = l-1 \text{ or } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= t_1 - (2^{l-3} + 2^{l-2}) \geq 0$ and $\mu = \left\lfloor \frac{t_1 - (2^{l-3} + 2^{l-2})}{2^{l-2}} \right\rceil$. There are two cases:

2.2.2.1. $\mu = 0$. In this case, we have $0 \leq \frac{t_1 - 2^{l-3} - 2^{l-2}}{2^{l-2}} < \frac{1}{2}$, i.e. $2^{l-3} + 2^{l-2} \leq t_1 < 2^{l-1}$. This implies $\boxed{t_{1,l-1} = 0}$. Therefore, condition (5.6) gives $\boxed{t_{1,l-2} = 1}$. Moreover, we have $2^{l-3} \leq t_1 - 2^{l-2} < 2^{l-2}$; hence $\boxed{t_{1,l-3} = 1}$.

2.2.2.2. $\mu = 1$. We have $2^{l-1} \leq t_1 \leq 2^l - 2$. This implies $\boxed{t_{1,l-1} = 1}$. We now have to recover $t_{1,l-2}, t_{1,l-3}$.

Next, query the decryption oracle with

$$\mathbf{y} = (0, \ldots, 0, -1, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

with -1 in l-th and $(l+1)$-th positions:

$$y_i = \begin{cases} -1 & \text{if } i = l \text{ or } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= t_1 - 2^{l-1} \geq 0$ and $\mu = \left\lfloor \frac{t_1 - 2^{l-1}}{2^{l-2}} \right\rceil$. There are two cases:

2.2.2.2.1. $\mu = 0$. In this case, we have $0 \leq \frac{t_1 - 2^{l-1}}{2^{l-2}} < \frac{1}{2}$, i.e. $0 \leq t_1 - 2^{l-1} < 2^{l-3} = \sum_{j=0}^{l-4} 2^j + 1$.

This implies $\boxed{t_{1,l-2} = t_{1,l-3} = 0}$.

2.2.2.2.2. $1 \leq \mu \leq 3$. In particular, $2^{l-3} \leq t_1 - 2^{l-1}$. Then we have

$$(t_{1,l-2}, t_{1,l-3}) \in \{0,1\}^2 \backslash \{(0,0)\} \qquad (5.7)$$

Next, query the decryption oracle with

$$\mathbf{y} = (0, \ldots, 0, -1, 0, -1, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

with -1 in $(l-2)$-th, l-th and $(l+1)$-th positions:

$$y_i = \begin{cases} -1 & \text{if } i = l-2, i = l \text{ or } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= t_1 - (2^{l-1} + 2^{l-3}) \geq 0$ and $\mu = \left\lfloor \frac{t_1 - 2^{l-1} - 2^{l-3}}{2^{l-2}} \right\rceil$. There are two cases:

2.2.2.2.2.1. $\mu = 0$. In this case, we have $0 \leq \frac{t_1 - 2^{l-1} - 2^{l-3}}{2^{l-2}} < \frac{1}{2}$, i.e. $2^{l-3} \leq t_1 - 2^{l-1} < 2^{l-2}$. This means that $\boxed{t_{1,l-2} = 0}$. Condition (5.7) then implies $\boxed{t_{1,l-3} = 1}$.

2.2.2.2.2.2. $1 \leq \mu \leq 2$. In particular, $2^{l-2} \leq t_1 - 2^{l-1} \leq 2^l - 2 - 2^{l-1} = 2^{l-1} - 2$. Then, we have $\boxed{t_{1,l-2} = 1}$. We still have to find $t_{1,l-3}$. Next, query the decryption oracle with $\mathbf{y} = (0, \ldots, 0, -1, -1, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$, where -1 is in $(l-1)$-th, l-th and $(l+1)$-th positions:

$$y_i = \begin{cases} -1 & \text{if } i = l-1, i = l \text{ or } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Through the decryption oracle, we have $x =< \mathbf{y}, \mathbf{v} >= t_1 - (2^{l-1} + 2^{l-2}) \geq 0$ and $\mu = \left\lfloor \frac{t_1 - 2^{l-1} - 2^{l-2}}{2^{l-2}} \right\rceil$. There are two cases:

2.2.2.2.2.2.1. $\mu = 0$. In this case,

$$0 \leq \frac{t_1 - 2^{l-1} - 2^{l-2}}{2^{l-2}} < \frac{1}{2}, \quad \text{i.e.}$$

i.e. $0 \leq t_1 - 2^{l-1} - 2^{l-2} < 2^{l-3}$. This implies that $\boxed{t_{1,l-3} = 0}$.

2.2.2.2.2.2.2. $\mu = 1$. Then $2^{l-3} \leq t_1 - 2^{l-1} - 2^{l-2}$. This implies that $\boxed{t_{1,l-3} = 1}$.

At this point, we know the first three significant bits $t_{1,l-1}, t_{1,l-2}, t_{1,l-3}$ of t_1. Notice that we have recovered the first three most significant bits with at most 7 oracle queries. Next, we are going to recover $t_{1,l-4}$. Query the decryption oracle with

$$\mathbf{y} = (0, \ldots, 0, -t_{1,l-3}, -t_{1,l-2}, -t_{1,l-1}, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

where $-t_{1,i}$ is in $(i+1)$-th position. Then

$$x =< \mathbf{y}, \mathbf{v} >= t_1 - (t_{1,l-1}2^{l-1} + t_{1,l-2}2^{l-2} + t_{1,l-3}2^{l-3})$$

Now, we have $0 \leq x < 2^{l-3}$. Therefore, $\mu = \lfloor x/2^{l-2} \rceil = 0$, and so not useful at all to learn $t_{1,l-4}$. The idea is to 'shift' the bits 'to the left', i.e. towards the most significant. So, let us instead choose

$$\mathbf{y} = 2 \cdot (0, \ldots, 0, -t_{1,l-3}, -t_{1,l-2}, -t_{1,l-1}, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

So now $x =< \mathbf{y}, \mathbf{v} >$ is such that $0 \leq x < 2^{l-2}$. After submitting $\mathbf{y}$ to the decryption oracle, it will compute and output $\mu = \lfloor x/2^{l-2} \rceil$. Then $\boxed{t_{1,l-4} = \mu}$.

Now we can generalize and recover $t_{1,k}$, for all $k = l-4, l-5, \ldots, 1, 0$. This will complete the recovery of t_1. Suppose that, for a given k, we recovered already $t_{1,m}$, $\forall m \in [k+1, \ldots, l-1]$. We then recover $t_{1,k}$ by recurrence. Choose

$$\mathbf{y} = 2^{l-k-3}(0, \ldots, 0, -t_{1,k+1}, -t_{1,k+2}, \ldots, -t_{1,l-1}, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

with $-t_{1,i}$ in $(i+1)$-th position; i.e.

$$y_i = \begin{cases} -2^{l-k-3}t_{1,i-1} & \text{for } i \in [k+2, \ldots, l] \\ -2^{l-k-3} & \text{for } i = l+1 \\ 0 & \text{otherwise} \end{cases}$$

Then we have $x =< \mathbf{y}, \mathbf{v} >= 2^{l-k-3}\left(t_1 - \sum_{j=k+1}^{l-1} t_{1,j}2^j\right)$ with $0 \leq x < 2^{l-2}$. Then, $\boxed{t_{1,k} = \mu}$.

We recover completely t_1 after at most $7 + (l-3) = l+4$ oracle queries.

Recovering $\mathsf{BitDecomp(t_r)}$, for every $r \in [1, 2, \ldots, n]$

We can now generalize and recover $\mathsf{BitDecomp(t_r)}$, for every $r \in [1, 2, \ldots, n]$, in a way analogous to what has been done for the case $r = 1$. The only difference is that, when choosing $\mathbf{y} \in \mathbb{Z}_q^N$, we set -1 in position $rl+1$. So, for a given $r \in [1, 2, \ldots, n]$, we have the following.

- Recovering the first three most significant bits $t_{r,l-1}, t_{r,l-2}, t_{r,l-3}$. This is done exactly as in the case of t_1, with the only modification $y_{l+1} = 0$ and $y_{rl+1} = -1$ always.

- Recovering $t_{r,k}$, for all $k = l-4, l-5, \ldots, 1, 0$. Suppose that, for a given k, we recovered already $t_{r,m}$, $\forall m \in [k+1, \ldots, l-1]$. We then recover $t_{r,k}$ by recurrence. Choose

$$\mathbf{y} = 2^{l-k-3}(0, \ldots, 0, -t_{r,k+1}, -t_{r,k+2}, \ldots, -t_{r,l-1}, 0, \ldots, 0, -1, 0, \ldots, 0) \in \mathbb{Z}_q^N$$

with $-t_{r,i}$ in $(i+1)$-th position and -1 in $(rl+1)$-th position; i.e.

$$y_i = \begin{cases} -2^{l-k-3} t_{r,i-1} & \text{for } i \in [k+2, \dots, l] \\ -2^{l-k-3} & \text{for } i = rl+1 \\ 0 & \text{otherwise} \end{cases}$$

Then we have $x =< \mathbf{y}, \mathbf{v} >= 2^{l-k-3} \left(t_r - \sum_{j=k+1}^{l-1} t_{r,j} 2^j \right)$ with $0 \leq x < 2^{l-2}$. Then, $\boxed{t_{r,k} = \mu}$.

In summary, we can recover the secret key $\mathbf{t} \in \mathbb{Z}_q^n$ with at most $(l+4) \cdot n$ oracle queries.

5.8.3 Algorithmic Description

Formally, the attack from Section 5.8.2 can be described by Algorithm 6. For a given vector $\mathbf{y} = (y_1, \dots, y_N) \in \mathbb{Z}_q^N$, we let $C_{\mathbf{y}} \in \mathcal{M}_{N \times N}(\mathbb{Z}_q)$ be the square $N \times N$ matrix whose entries are all 0 except for the $(l-2)$-th row C_{l-2}, which is $\mathbf{y}$. We have denoted the decryption oracle $\mathcal{O}_D(C_{\mathbf{y}}) := \mathsf{Decrypt}(\mathsf{sk}, C_{\mathbf{y}}) = \left\lfloor \frac{<\mathbf{y},\mathbf{v}>}{2^{l-2}} \right\rceil$. For ease of notation, we have also considered the standard vectors $\mathbf{e}_1, \dots, \mathbf{e}_N \in \mathbb{Z}_q^N$: for every $i = 1, \dots, N$, $\mathbf{e}_i$ is the $\mathbf{0}$-vector except in position i, where it has value 1:

$$\mathbf{e}_i = (e_{i,1}, \dots, e_{i,N}) = (0, \dots, 0, 1, 0, \dots, 0), \quad e_{i,i} = 1, e_{i,j} = 0 \text{ for } j \neq i$$

We put $\mathbf{d}_i := -\mathbf{e}_i$, for all i.

Algorithm 6 Key Recovery Attack against GSW13 SHE

```
input: $q, n$
$l \leftarrow \lfloor \log_2 q \rfloor + 1; \quad N \leftarrow (n+1) \cdot l$
for $r = 1$ to $n$ do
  $\mathbf{y} \leftarrow \mathbf{d}_{rl+1}$
  if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
    $t_{r,l-1}, t_{r,l-2}, t_{r,l-3} \leftarrow 0$
  else
    $\mathbf{y} \leftarrow \mathbf{d}_{l-2} + \mathbf{d}_{rl+1}$
    if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
      $t_{r,l-1}, t_{r,l-2} \leftarrow 0; \quad t_{r,l-3} \leftarrow 1$
    else
      $\mathbf{y} \leftarrow \mathbf{d}_{l-1} + \mathbf{d}_{rl+1}$
      if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
        $t_{r,l-1}, t_{r,l-3} \leftarrow 0; \quad t_{r,l-2} \leftarrow 1$
      else
        $\mathbf{y} \leftarrow \mathbf{d}_{l-2} + \mathbf{d}_{l-1} + \mathbf{d}_{rl+1}$
        if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
          $t_{r,l-1} \leftarrow 0; \quad t_{r,l-2}, t_{r,l-3} \leftarrow 1$
        else
          $t_{r,l-1} \leftarrow 1; \quad \mathbf{y} \leftarrow \mathbf{d}_l + \mathbf{d}_{rl+1}$
          if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
            $t_{r,l-2}, t_{r,l-3} \leftarrow 0$
          else
            $\mathbf{y} \leftarrow \mathbf{d}_{l-2} + \mathbf{d}_l + \mathbf{d}_{rl+1}$
            if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
              $t_{r,l-2} \leftarrow 0; \quad t_{r,l-3} \leftarrow 1$
            else
              $t_{r,l-2} \leftarrow 1; \quad \mathbf{y} \leftarrow \mathbf{d}_{l-1} + \mathbf{d}_l + \mathbf{d}_{rl+1}$
              if $\mathcal{O}_D(C_{\mathbf{y}}) = 0$ then
                $t_{r,l-3} \leftarrow 0$
              else
                $t_{r,l-3} \leftarrow 1$
              end if
            end if
          end if
        end if
      end if
    end if
  end if
  for $k = l-4$ to $0$ do
    $\mathbf{y} \leftarrow 2^{l-k-3} \cdot (\mathbf{d}_{rl+1} + \sum_{i=k+2}^{l} t_{r,i-1}\mathbf{d}_i)$
    $t_{r,k} \leftarrow \mathcal{O}_D(C_{\mathbf{y}})$
  end for
end for
```

```
for $i = 1$ to $n$ do
    $t_i \leftarrow \mathsf{BitDecomp}^{-1}(t_{i,0}, t_{i,1}, \ldots, t_{i,l-1})$
end for
$\mathbf{t} \leftarrow (t_1, \ldots, t_n)$
return $\mathbf{t}$
```

We now focus on the NTRU-based SHE schemes. Gentry's original FHE scheme is based on ideal lattices and it is implemented using cyclotomic rings. NTRU is a practical lattice-based cryptosystem, which is also based on cyclotomic rings, that remained without a security proof for a long time. NTRU was recently put on a stronger foundation by Stehle and Steinfeld [74], and NTRU-based cryptosystems returned to become an interesting research area. Scale-invariant homomorphic encryption was proposed by Brakerski [11], presenting a construction that avoids the utilization of modulus switching technique, considerably simplifying the scheme. We now present original adaptive key recovery attacks on NTRU-based SHE schemes. In particular, we attack the scale-invariant proposal [9, 55].

5.9 Attack against the LTV12 SHE Scheme

We start by recalling the LTV12 SHE Scheme [55]. Let λ be the security parameter, consider an integer $n = n(\lambda)$ and a prime number $q = q(\lambda) \neq 2$. Consider also a degree-n polynomial $\phi(x) = \phi_\lambda(x)$: following [55], we will use $\phi(x) = x^n + 1$. Finally, let $\chi = \chi(\lambda)$ a $B(\lambda)$-bounded error distribution over the ring $R := \mathbb{Z}[x]/(\phi(x))$. The parameters $n, q, \phi(x)$ and χ are public and we assume that given λ, there are polynomial-time algorithms that output n, q and $\phi(x)$, and sample from the error distribution χ. The message space is $\mathcal{M} = \{0, 1\}$, and all operations on ciphertexts are carried out in the ring $R_q := \mathbb{Z}_q[x]/(\phi(x))$.

$\mathsf{KeyGen}(\lambda)$:

- sample $f', g \leftarrow \chi$
- set $f := 2f' + 1$ so that
  $f \equiv 1 \bmod 2$
- if f is not invertible in R_q,
  resample f'
- $\mathsf{pk} := h = 2gf^{-1} \in R_q$
- $\mathsf{sk} := f \in R$

$\mathsf{Encrypt}(\mathsf{pk}, m)$:

- sample $s, e \leftarrow \chi$
- output ciphertext $c := hs + 2e + m \in R_q$

$\mathsf{Decrypt}(\mathsf{sk}, c)$:

- let $\mu = f \cdot c \in R_q$
- output $\mu' := \mu \bmod 2$

Since we don't need the evaluation step, we omit it in the description. In the original paper [55], the somewhat homomorphic encryption scheme is multi-key, i.e. one can use several secret keys $\mathsf{sk}_1 = f_1, \ldots, \mathsf{sk}_M = f_M$ in order to decrypt. By analyzing the original decryption step, one can see that, in order to decrypt the plaintext message, we need to multiply secret keys $\mathsf{sk}_1 = f_1, \ldots, \mathsf{sk}_M = f_M$ together, and then multiply the result with the ciphertext and reduce. For this reason, it is enough to retrieve, as the secret key, the polynomial $f_1 \cdots f_M =: s = s(x) = s_0+s_1x+s_2x^2+\cdots+s_{n-1}x^{n-1} \in R_q$, with $s_i \in (-q/2, q/2]$ for all $i = 0, 1, \ldots, n-1$. For this reason, it is enough to present the scheme as we saw it, with only one secret key.

Remark 5. In [55], the authors do not explicitly state how the decryption behaves if $\mu \bmod 2$ is not a constant. We consider three scenarios: (1) output directly $\mu \bmod 2$; (2) output the constant of $\mu \bmod R_2$; (3) output an error. In the following, we describe here a key recovery attack for scenario (1) and it can be easily extended to scenario (2), which we will show later on in Chapter 6. It is likely that we can adapt our attack to scenario (3), but we have not succeeded so far. We will include a discussion about scenario (3) in Chapter 6.

5.9.1 Attack Preview

Generally, suppose the secret key is in the form of the polynomial $f = s(x) = s_0 + s_1x + s_2x^2 + \cdots + s_{n-1}x^{n-1} \in R_q$. Now, since we assume q odd, and s_i is an integer, we have $-q/2 < s_i < q/2$, and in particular $-\left\lfloor \frac{q}{2} \right\rfloor \leq s_i \leq \left\lfloor \frac{q}{2} \right\rfloor, \quad \forall 0 \leq i \leq n-1$. Each coefficient s_i can have $\left\lfloor \frac{q}{2} \right\rfloor - (-\left\lfloor \frac{q}{2} \right\rfloor) + 1 = q$ possible different values. We remark that there exists a bit representation of the s_i's such that $\#\text{bits}(s_i) = \lfloor \log_2(q-1) \rfloor + 1 =: N$, and $\#\text{bits}(s) = n \cdot \#\text{bits}(s_i) = n \cdot (\lfloor \log_2(q-1) \rfloor + 1)$. The decryption oracle reveals a polynomial $\mu'(x) = \mu(x) \bmod 2 = \mu'_0 + \mu'_1 x + \cdots + \mu'_{n-1}x^{n-1}$, with $\mu'_i \in \{0,1\}$ for $i = 0, 1, \ldots, n-1$. Hence, decryption oracle reveals n bits at a time. Therefore, the minimum number of oracle queries needed to recover s is N. As we will see, our attack needs N oracle queries, plus at most $n-1$ oracle queries necessary to determine the signs of the coefficients of the secret key. We remark that the scheme as described in [55] has message space $\mathcal{M} = \{0,1\}$. When the oracle decryption receives an honestly-generated ciphertext, it returns either $0 = \sum_{i=0}^{n-1} 0 \cdot x^i \in R_q$ or $1 = 1 + \sum_{i=1}^{n-1} 0 \cdot x^i \in R_q$. However, in principle the oracle decryption can return any polynomial in $\{0,1\}/(x^n+1)$ and we will use this fact as basis to build our attack.

Here is the workflow of our key recovery attack. First of all, we are going to determine the parity of each coefficient $s_i \in (-q/2, q/2]$. Then, we are going to find s_i by gradually reducing (halving) the interval in which it lies. At some point, s_i will be reduced to belong to some interval with at most two consecutive integers; the absolute value of s_i will be deduced by its (known) parity. At this point, we will know the secret key coefficient s_i in absolute value; in the last step, we are going to query the oracle decryption at most n times in order to recover the sign of the coefficients s_i, for $i = 1, 2, \ldots, n-1$, relative to the (unknown) sign of s_0. So in the end, we will end up with two possible candidate secret keys $s_1(x)$ and $s_2(x) = -s_1(x)$. We have then $s(x) = s_1(x)$ or $s(x) = s_2(x)$, and recovering which one of the two is trivial with an extra oracle query.

In our description, we consider the coefficients s_i in the interval $(-q/2, q/2]$ and can recover the private key with at most $\lfloor \log_2 q \rfloor + n$ decryption oracle queries. However, we could consider the stricter interval $[-B, B]$, with B the bound on coefficients given by the distribution χ from which the coefficients are picked from. In this case, we can see that the total number of queries needed to be submitted to the decryption oracle are actually at most $\lfloor \log_2 B \rfloor + n$.

5.9.2 Detailed Attack

Preliminary Step

Submit to the decryption oracle the “ciphertext" $c(x) = 1 \in R_q$. The oracle will compute and return the polynomial $D(c(x) = 1) = s(x) \bmod 2 = \sum_{i=0}^{n-1}(s_i \bmod 2)x^i$, which tells us the parity of each s_i, $i = 0, 1, \ldots, n-1$.

Step 1.

Choose and submit to the decryption oracle the “ciphertext" $c(x) = 2 \in R_q$. It will compute and return the polynomial $D(c(x) = 2) = (2s(x) \in R_q) \bmod 2 = \sum_{i=0}^{n-1}\left[(2s_i \bmod q) \bmod 2\right] x^i$. For all $i \in [0, n-1]$ we have

$$\frac{-q+1}{2} \leq s_i \leq \frac{q-1}{2}, \text{ and so } -q+1 \leq 2s_i \leq q-1 \tag{A}$$

For each i, we have two cases to distinguish:

Case A_1: $(2s_i \bmod q) \bmod 2 = 0$. Then, condition (A) implies that $\frac{-q+1}{2} \leq 2s_i \leq \frac{q-1}{2}$, i.e. $\frac{-q+1}{4} \leq s_i \leq \frac{q-1}{4}$

$$-q+1 \leq 4s_i \leq q-1 \tag{A1}$$

Case B_1: $(2s_i \bmod q) \bmod 2 = 1$. Then, condition (A) implies that $\frac{q-1}{2} + 1 \leq 2|s_i| \leq q-1$, i.e. $\frac{q+1}{4} \leq |s_i| \leq \frac{q-1}{2}$

$$q+1 \leq 4|s_i| \leq 2q-2 \tag{B1}$$

Step 2.

Choose and submit to the decryption oracle the "ciphertext" $c(x) = 4 \in R_q$. It will compute and return the polynomial $D(c(x) = 4) = [s(x) \cdot 4]_q \bmod 2 = \sum_{i=0}^{n-1} [[4s_i]_q \bmod 2]\, x^i$. For each i, we have four cases to distinguish:

Case A_2: In Step 1 case A_1 held, and $[4s_i]_q \bmod 2 = 0$. Then, condition (A1) implies that $\frac{-q+1}{2} \leq 4s_i \leq \frac{q-1}{2}$, i.e. $\frac{-q+1}{8} \leq s_i \leq \frac{q-1}{8}$

$$-q+1 \leq 8s_i \leq q-1 \tag{A2}$$

Case B_2: In Step 1 case A_1 held, and $[4s_i]_q \bmod 2 = 1$. Then, condition (A1) implies that $\frac{q-1}{2} + 1 \leq 4|s_i| \leq q-1$, i.e. $\frac{q+1}{8} \leq |s_i| \leq \frac{q-1}{4}$

$$q+1 \leq 8|s_i| \leq 2q-2 \tag{B2}$$

Case C_2: In Step 1 case B_1 held, and $[4s_i]_q \bmod 2 = 0$. Then, condition (B1) implies that $q+1+\frac{q-1}{2} \leq 4|s_i| \leq 2q-2$, i.e. $\frac{3q+1}{8} \leq |s_i| \leq \frac{q-1}{2}$

$$3q+1 \leq 8|s_i| \leq 4q-4 \tag{C2}$$

Case D_2: In Step 1 case B_1 held, and $[4s_i]_q \bmod 2 = 1$. Then, condition (B1) implies that $q+1 \leq 4|s_i| \leq \frac{3q-1}{2}$, i.e. $\frac{q+1}{4} \leq |s_i| \leq \frac{3q-1}{8}$

$$2q+2 \leq 8|s_i| \leq 3q-1 \tag{D2}$$

Step 3.

Choose and submit to the decryption oracle the “ciphertext" $c(x) = 8 \in R_q$. It will compute and return the polynomial $D(c(x) = 8) = [s(x) \cdot 8]_q \bmod 2 = \sum_{i=0}^{n-1} \left[[8s_i]_q \bmod 2 \right] x^i$. For each i, we have four cases to distinguish:

Case A_3: In Step 2 case A_2 held, and $[8s_i]_q \bmod 2 = 0$. Then, condition (A2) implies that $\frac{-q+1}{2} \leq 8s_i \leq \frac{q-1}{2}$, i.e. $\frac{-q+1}{16} \leq s_i \leq \frac{q-1}{16}$

$$-q + 1 \leq 16s_i \leq q - 1 \tag{A3}$$

Case B_3: In Step 2 case A_2 held, and $[8s_i]_q \bmod 2 = 1$. Then, condition (A2) implies that $\frac{q-1}{2} + 1 \leq 8|s_i| \leq q - 1$, i.e. $\frac{q+1}{16} \leq |s_i| \leq \frac{q-1}{8}$

$$q + 1 \leq 16|s_i| \leq 2q - 2 \tag{B3}$$

Case C_3: In Step 2 case B_2 held, and $[8s_i]_q \bmod 2 = 0$. Then, condition (B2) implies that $\frac{3q+1}{2} \leq 8|s_i| \leq 2q - 2$, i.e. $\frac{3q+1}{16} \leq |s_i| \leq \frac{q-1}{4}$

$$3q + 1 \leq 16|s_i| \leq 4q - 4 \tag{C3}$$

Case D_3: In Step 2 case B_2 held, and $[8s_i]_q \bmod 2 = 1$. Then, condition (B2) implies that $q + 1 \leq 8|s_i| \leq \frac{3q-1}{2}$, i.e. $\frac{q+1}{8} \leq |s_i| \leq \frac{3q-1}{16}$

$$2q + 2 \leq 16|s_i| \leq 3q - 1 \tag{D3}$$

Case E_3: In Step 2 case C_2 held, and $[8s_i]_q \bmod 2 = 0$. Then, condition (C2) implies that $\frac{7q+1}{2} \leq 8|s_i| \leq 4q-4$, i.e. $\frac{7q+1}{16} \leq |s_i| \leq \frac{q-1}{2}$

$$7q+1 \leq 16|s_i| \leq 8q-8 \tag{E3}$$

Case F_3: In Step 2 case C_2 held, and $[8s_i]_q \bmod 2 = 1$. Then, condition (C2) implies that $3q+1 \leq 8|s_i| \leq \frac{7q-1}{2}$, i.e. $\frac{3q+1}{8} \leq |s_i| \leq \frac{7q-1}{16}$

$$6q+2 \leq 16|s_i| \leq 7q-1 \tag{F3}$$

Case G_3: In Step 2 case D_2 held, and $[8s_i]_q \bmod 2 = 0$. Then, condition (D2) implies that $2q+2 \leq 8|s_i| \leq \frac{5q-1}{2}$, i.e. $\frac{q+1}{4} \leq |s_i| \leq \frac{5q-1}{16}$

$$4q+4 \leq 16|s_i| \leq 5q-1 \tag{G3}$$

Case H_3: In Step 2 case D_2 held, and $[8s_i]_q \bmod 2 = 1$. Then, condition (D2) implies that $\frac{5q+1}{2} \leq 8|s_i| \leq 3q-1$, i.e. $\frac{5q+1}{16} \leq |s_i| \leq \frac{3q-1}{8}$

$$5q+1 \leq 16|s_i| \leq 6q-2 \tag{H3}$$

Final step.

We continue in this fashion and finally we obtain integers $s_i' := |s_i| \in [0, \frac{q-1}{2}]$, for $i = 0, 1, \ldots, n-1$. This is obtained in the last step, where all coefficients $|s_i|$, in absolute value, can assume at most only two (consecutive) values; the known parity will then determine $|s_i|$. It is easy to see that in order to achieve this we need $\lfloor \log_2 q \rfloor$ steps.

The strategy now is to find out whether $s_i \cdot s_j < 0$ or $s_i \cdot s_j > 0$ holds, for every i, j with $s_i, s_j \neq 0$. Let s_m be the first non-zero coefficient. This way, we will obtain two possible candidates of the secret key, one with $s_m > 0$ and the other with $s_m < 0$. A trivial query to the oracle decryption will allow us to determine which is the correct secret key.

We have to choose an appropriate “ciphertext" $c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1}$ to submit to the decryption oracle. Choose $c_0 = 1, c_1 = 1$ and $c_j = 0$ for $j \neq 0, 1$. Oracle decryption will compute and return the polynomial

$$D(c(x)) = s(x) \cdot c(x) = [s_0 - s_{n-1}]_q \bmod 2 + \sum_{i=1}^{n-1}([s_i + s_{i-1}]_q \bmod 2)x^i$$

Fix $i = 1, 2, \ldots, n-1$ such that $s_i, s_{i-1} \neq 0$. Let $b_i := [s_i + s_{i-1}]_q \bmod 2$ be the coefficient of x^i, and let $b'_i := [s'_i + s'_{i-1}]_q \bmod 2$. There are two cases to consider:

- $s'_i + s'_{i-1} \geq \frac{q+1}{2}$. Then
  - if $b_i = b'_i$, then s_i and s_{i-1} have the same sign;
  - if $b_i \neq b'_i$, then s_i and s_{i-1} have different signs.
- $0 \leq s'_i + s'_{i-1} \leq \frac{q-1}{2}$. Then we need to make an extra query to understand whether s_i and s_{i-1} have the same sign or not.

Now, for each one of the i of the previous case (i.e. such that $0 \leq s'_i + s'_{i-1} \leq \frac{q-1}{2}$, $i = 1, 2, \ldots, n-1$, and $s_i, s_{i-1} \neq 0$) we choose and submit to the decryption oracle the polynomial $c(x) = \alpha_i |s_{i-1}| + \alpha_i |s_i| x$, i.e. we choose $c_0 = \alpha_i |s_{i-1}|$, $c_1 = \alpha_i |s_i|$, $c_2 = c_3 = \cdots = c_{n-1} = 0$, where α_i is chosen such that

$$\alpha_i |s_{i-1} \cdot s_i| \in \left(\frac{q-1}{4}, \frac{q-1}{2} \right] \tag{5.8}$$

(it is always possible to find such an α_i). The oracle decryption will return the polynomial

$$D(c(x)) = s(x) \cdot c(x) \;=\; [\alpha_i|s_{i-1}|s_0 - \alpha_i|s_i|s_{n-1}]_q \bmod 2 + \\ + \sum_{j=1}^{n-1} \left([\alpha_i|s_{i-1}|s_j + \alpha_i|s_i|s_{j-1}]_q \bmod 2\right) x^j$$

Let's focus on the coefficient of x^i, i.e. $\beta_i := [\alpha_i|s_{i-1}|s_i + \alpha_i|s_i|s_{i-1}]_q \bmod 2$. Now, there are two cases:

- if s_i, s_{i-1} have different signs, then $\beta_i = 0$;
- if s_i, s_{i-1} have the same sign, then $\beta_i = 1$ (trivial to verify: 5.8 holds, and therefore $[2\alpha_i \cdot |s_i \cdot s_{i-1}|]_q$) is odd.

By repeating this idea for every $i = 1, 2, \ldots, n-1$ such that $0 \leq s'_i + s'_{i-1} \leq \frac{q-1}{2}$ we will know which one of the following relations $s_i \cdot s_{i-1} < 0 \quad \vee \quad s_i \cdot s_{i-1} > 0$ holds, for every consecutive non-zero coefficients s_i, s_{i-1}.

Now, we have one more thing to consider: we have to be careful in case one of the coefficient s_i is zero. In this case in fact, no information can be given about the sign of s_{i-1} if we compare it to s_i. To solve this problem, we have to choose and submit to the decryption oracle a polynomial $c(x) = a + bx^j$ for appropriates a, b, j. Let $0 \leq m_1 \leq n-1$ be an integer such that s_{m_1} is the first non-zero coefficient of the secret key $s(x)$. If there exists $i_1 > m_1$ such that $s_{i_1} = 0$, then let m_2 be the first non-zero coefficient such that $i_1 < m_2 \leq n-1$. Then we want to compare the relative signs of s_{m_1} and s_{m_2} by choosing the polynomial $c(x)$ with $c_0 = \alpha|s_{m_1}|$, $c_{m_2-m_1} = \alpha|s_{m_2}|$, $c_j = 0$ for $j \neq 0, m_2 - m_1$. So we have $c(x) = \alpha|s_{m_1}| + \alpha|s_{m_2}|x^{m_2-m_1}$, with α such that $\alpha|s_{m_1}s_{m_2}| \in \left(\frac{q-1}{4}, \frac{q-1}{2}\right]$. The oracle decryption will return the polynomial $D(c(x)) = s(x) \cdot c(x) = \beta_0 + \beta_1 x + \cdots + \beta_{n-1}x^{n-1}$. Consider the m_2-th coefficient $\beta_{m_2} = [\alpha|s_{m_1}|s_{m_2} + \alpha|s_{m_2}|s_{m_1}]_q \bmod 2$. As before, we can conclude that if s_{m_1}, s_{m_2} have different signs, then $\beta_{m_2} = 0$, and if s_{m_1}, s_{m_2} have the same sign, then $\beta_{m_2} = 1$.

Now, similar to what just discussed, if there exists $i_2 > m_2$ such that $s_{i_2} = 0$, then let m_3 be the first non-zero coefficient such that $m_3 > i_2$. We will in a similar fashion compare the relative signs of s_{m_1} and s_{m_3}. We keep proceeding this way, and in the end we will know, for every $0 \leq i, j \leq n-1$ such that $s_i \neq 0, s_j \neq 0$, whether $s_i \cdot s_j > 0$ or $s_i \cdot s_j < 0$ occurs. This allows us to determine two possible candidates for the secret key $s(x)$ (assume s_m is the first non-zero coefficient; then one candidate has $s_m < 0$, the other has $s_m > 0$). A trivial oracle decryption query will reveal which one of the two is the correct secret key. The total number of queries needed to be submitted to the oracle decryption query is then at most $\lfloor \log_2 q \rfloor + n$.

5.10 Attack against the BLLN13 SHE Scheme

We start by recalling the BLLN13 SHE Scheme [9]. For a given positive integer $d \in \mathbb{N}_{>0}$, define the quotient ring $R := \mathbb{Z}[x]/(\Phi_d(x))$, i.e. the ring of polynomials with integer coefficients modulo the d-th cyclotomic polynomial $\Phi_d(x) \in \mathbb{Z}[x]$. The degree of Φ_d is $n = \varphi(d)$, where φ is Euler's totient function. As considered by the authors of [9], for correctness of the scheme, let d be a power of 2; in this case, we have $\Phi_d(x) = x^n + 1$ with n also a power of 2. Therefore $R = \mathbb{Z}[x]/(x^n + 1)$. The other parameters of the [9] SHE scheme are a prime integer $q \in \mathbb{N}$ and an integer $t \in \mathbb{N}$ such that $1 < t < q$. Let also $\chi_{\text{key}}, \chi_{\text{err}}$ be two distributions on R. The parameters $d, q, t, \chi_{\text{key}}$ and χ_{err} are public and we assume that given λ, there are polynomial-time algorithms that output d, q, t and

$\phi(x)$, and sample from the error distributions χ. The message space is $\mathcal{M} = R/tR = \mathbb{Z}_t[x]/(x^n + 1)$, and all operations on ciphertexts are carried out in the ring $R_q := \mathbb{Z}_q[x]/(\phi(x))$.

$\mathsf{KeyGen}(\lambda)$:

- sample $f', g \leftarrow \chi_{\text{key}}$
- let $f = [tf' + 1]_q$
- if f is not invertible in R_q, resample f'
- set $\mathsf{pk} := h = [tgf^{-1}]_q \in R_q$
- set $\mathsf{sk} := f \in R_q$

$\mathsf{Encrypt}(\mathsf{pk}, m)$**:**

- for a message $m + tR$, choose $[m]_t$ as its representative
- sample $s, e \leftarrow \chi_{\text{err}}$
- output ciphertext $c = [\lfloor q/t \rfloor [m]_t + e + hs]_q \in R_q$

$\mathsf{Decrypt}(\mathsf{sk}, c)$**:**

- output $m = \left[\left\lfloor \frac{t}{q} \cdot [fc]_q \right\rceil\right]_t \in R_t$

Since we don't need the evaluation step, we omit it in the description.

5.10.1 Attack Preview

We are going to recover the secret key $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_{n-1}x^{n-1} \in \frac{\mathbb{Z}_q[x]}{(x^n+1)}$, where f_i is an integer in $(-q/2, q/2]$ for all $i = 0, 1, \ldots, n-1$. In order to recover $f(x)$, we are going to submit specifically-chosen 'ciphertexts' of the form $c(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1}x^{n-1} \in$

$\frac{\mathbb{Z}_q[x]}{(x^n+1)}$, with integers $c_i \in (-q/2, q/2]$. Choose $c(x) = 1 = 1 + 0x + 0x^2 + \cdots + 0x^{n-1}$. We have

$$\begin{aligned} D(c=1) &= \left[\left\lfloor \frac{t}{q} \cdot [f \cdot 1]_q \right\rceil\right]_t \\ &= \left[\left\lfloor \frac{t}{q} \cdot \left([f_0]_q + [f_1]_q x + [f_2]_q x^2 + \cdots + [f_{n-1}]_q x^{n-1}\right) \right\rceil\right]_t \\ &\overset{*}{=} \left[\left\lfloor \frac{t}{q} \cdot \left(f_0 + f_1 x + \cdots + f_{n-1} x^{n-1}\right) \right\rceil\right]_t \\ &= \left[\left\lfloor \frac{t}{q} f_0 \right\rceil + \left\lfloor \frac{t}{q} f_1 \right\rceil x + \cdots + \left\lfloor \frac{t}{q} f_{n-1} \right\rceil x^{n-1}\right]_t \end{aligned}$$

Equality $\overset{*}{=}$ holds since the integer coefficients f_i are already reduced modulo q. Now, for every $0 \leq i \leq n-1$ we have $-q/2 < f_i \leq q/2$. We have that $q > 2$ since in [9] it is claimed that $1 < t < q$, with t, q integers. In particular, q is a prime integer greater than 2, and therefore $q/2 \notin \mathbb{N}$. So we have $-q/2 < f_i < q/2$. In particular we have that $-\frac{t}{2} < \frac{t}{q} \cdot f_i < \frac{t}{2}$. For every $0 \leq i \leq n-1$, let $u_i^{(1)} := \left\lfloor \frac{t}{q} f_i \right\rceil$. We have $\left\lceil -\frac{t}{2} \right\rceil \leq u_i^{(1)} \leq \left\lfloor \frac{t}{2} \right\rfloor$. Each $u_i^{(1)}$ can have

$$\left\lfloor \frac{t}{2} \right\rfloor - \left\lceil -\frac{t}{2} \right\rceil + 1 = 2\left\lfloor \frac{t}{2} \right\rfloor + 1 = \begin{cases} t & \text{if } t \text{ is odd} \\ t+1 & \text{if } t \text{ is even} \end{cases}$$

possible different values, i.e. $u_i^{(1)}$ can have t different possible values if t is odd, and can have $t+1$ different possible values if t is even. Now, for every $0 \leq i \leq n-1$, we have that $[u_i^{(1)}]_t \in (-t/2, t/2]$ and therefore

- $[u_i^{(1)}]_t \in= [-\frac{t}{2} + \frac{1}{2}, -\frac{t}{2} + \frac{3}{2}, -\frac{t}{2} + \frac{5}{2}, \cdots, \frac{t}{2} - \frac{1}{2}] =: T_1$ if t is odd;
- $[u_i^{(1)}]_t \in \left[-\frac{t}{2} + 1, -\frac{t}{2} + 2, \ldots, \frac{t}{2}\right] =: T_2$ if t is even.

We have that $\#(T_1) = \#(T_2) = t$. Let $v_i^{(1)} := [u_i^{(1)}]_t$ for $0 \le i \le n-1$. It is clear that if $u_i^{(1)} = -t/2$, i.e. if $u_i^{(1)} = \lceil -t/2 \rceil$ and t is even, then $v_i^{(1)} = t/2$. We have

$$\begin{aligned} D(c(x) = 1) &= \left[u_0^{(1)} + u_1^{(1)}x + u_2^{(1)}x^2 + \cdots + u_{n-1}^{(1)}x^{n-1}\right]_t \\ &= [u_0^{(1)}]_t + [u_1^{(1)}]_t x + \cdots + [u_{n-1}^{(1)}]_t x^{n-1} \\ &= v_0^{(1)} + v_1^{(1)}x + v_2^{(1)}x^2 + \cdots + v_{n-1}^{(1)}x^{n-1} \end{aligned}$$

where, $\forall i = 0, 1, \ldots, n-1$, we have

$$v_i^{(1)} = \begin{cases} \frac{t}{2} & \text{if } u_i^{(1)} = -\frac{t}{2} (\text{i.e. if } u_i^{(1)} = \left\lceil -\frac{t}{2} \right\rceil \text{ and } t \text{ is even}) \\ u_i^{(1)} & \text{otherwise} \end{cases}$$

In particular, if t is odd, then $D(c = 1) = u_0^{(1)} + u_1^{(1)}x + u_2^{(1)}x^2 + \cdots + u_{n-1}^{(1)}x^{n-1}$.

We have, $\forall 0 \le i \le n-1$,

$$\text{if } t \text{ is odd, } -\frac{t}{2} + \frac{1}{2} \le v_i^{(1)} \le \frac{t}{2} - \frac{1}{2}; \text{ if } t \text{ is even, } -\frac{t}{2} + 1 \le v_i^{(1)} \le \frac{t}{2}$$

In both cases, $v_i^{(1)}$ can only have t different values. As we saw before, in case of t odd we need to perform $\lceil log_2(q/t) \rceil + 1$ oracle decryption queries; in case of t even, we need to perform extra oracle decryption queries (at most $n-1$) in order to understand which sign are given the coefficients of the secret key. Therefore, the total number of queries to the decryption oracle is at most $\lceil log_2(q/t) \rceil + n$. If we use the actual bound B given on the coefficients s_i by the distribution χ, we have that the total number of queries to the decryption oracle is at most $\lceil log_2(B/t) \rceil + n$.

5.10.2 Detailed Attack in three Cases

Case 1: t is odd

Step 1: select $c(x) = 1$

Select "ciphertext" $c(x) = 1$ and submit it to the decryption oracle. Since t is odd and $v_i^{(1)} = u_i^{(1)}$, $\forall 0 \leq i \leq n-1$, we obtain the polynomial $D(c = 1) = u_0^{(1)} + u_1^{(1)}x + u_2^{(1)}x^2 + \cdots + u_{n-1}^{(1)}x^{n-1}$, where $\left\lceil -\frac{t}{2} \right\rceil \leq u_i^{(1)} \leq \left\lfloor \frac{t}{2} \right\rfloor$. Every $u_i^{(1)}$ can have only t different values and can be written as $u_i^{(1)} = \left\lceil -\frac{t}{2} \right\rceil + k_{i,1}$, with $k_{i,1} \in \{0, 1, \ldots, t-1\}$. Now, it is easy to see that

$$u_i^{(1)} = \left\lceil -\frac{t}{2} \right\rceil + k_{i,1} \Leftrightarrow -\frac{q}{2} + \frac{q}{t}k_{i,1} < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1} + 1)$$

The polynomial obtained from the decryption oracle can therefore be written as

$$D(c(x) = 1) = u_0^{(1)} + u_1^{(1)}x + u_2^{(1)}x^2 + \cdots + u_{n-1}^{(1)}x^{n-1} = \sum_{i=0}^{n-1} \left(\left\lceil -\frac{t}{2} \right\rceil + k_{i,1} \right) x^i$$

Each f_i belongs to the interval $(-q/2, q/2)$. But after this our first query we learn values $k_{i,1} \in [0, 1, \ldots, t-1]$, $0 \leq i \leq n-1$, such that

$$-\frac{q}{2} + \frac{q}{t}k_{i,1} < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1} + 1) \qquad \text{(F(0,1))}$$

We have $-\frac{q}{2} + \frac{q}{t}(k_{i+1} + 1) - \left(-\frac{q}{2} + \frac{q}{t}k_{i+1} \right) = \frac{q}{t}$. Therefore, we know each integer coefficient f_i with an error up to $\frac{q}{t}$. The idea now is to keep submitting 'ciphertext' to the decryption oracle and obtain values $k_{i,j}$, with $0 \leq i \leq n-1$ and increasing integers $j = 1, 2, 3, \ldots$, in such a way that we keep reducing the interval in which f_i lies until we know f_i with an error smaller than 1, which determines each f_i completely.

Step 2: select $c(x) = 2$

Select now "ciphertext" $c(x) = 2 = 2 + 0x + 0x^2 + \cdots + 0x^{n-1}$. Decryption oracle computes and return the polynomial

$$\begin{aligned} D(c=2) &= \left[\left\lfloor \frac{t}{q} \cdot [f \cdot 2]_q \right\rceil\right]_t \\ &= \left[\left\lfloor \frac{t}{q} \cdot \left([2f_0]_q + [2f_1]_q x + [2f_2]_q x^2 + \cdots + [2f_{n-1}]_q x^{n-1}\right)\right\rceil\right]_t \\ &= \left[\left\lfloor \frac{t}{q} f_0^{(2)} \right\rceil + \left\lfloor \frac{t}{q} f_1^{(2)} \right\rceil x + \cdots + \left\lfloor \frac{t}{q} f_{n-1}^{(2)} \right\rceil x^{n-1}\right]_t \end{aligned}$$

where we have put $f_i^{(2)} := [2f_i]_q$, for every $0 \le i \le n-1$; of course we have $-\frac{q}{2} < f_i^{(2)} < \frac{q}{2}$. Now,

- if $-q/4 < f_i < q/4$, then $-\frac{q}{2} < 2f_i < \frac{q}{2}$ and therefore $f_i^{(2)} = [2f_i]_q = 2f_i$
- if $-q/2 < f_i < -q/4$, then $-q < 2f_i < -\frac{q}{2}$ and therefore $f_i^{(2)} = [2f_i]_q = 2f_i + q$
- if $q/4 < f_i < q/2$, then $\frac{q}{2} < 2f_i < q$ and therefore $f_i^{(2)} = [2f_i]_q = 2f_i - q$

So we have

$$f_i^{(2)} = [2f_i]_q = \begin{cases} 2f_i & \text{if } -\frac{q}{4} < f_i < \frac{q}{4} \\ 2f_i + q & \text{if } -\frac{q}{2} < f_i < -\frac{q}{4}, \text{ and in this case } 0 < f_i^{(2)} < \frac{q}{2} \\ 2f_i - q & \text{if } \frac{q}{4} < f_i < \frac{q}{2}, \text{ and in this case } -\frac{q}{2} < f_i^{(2)} < 0 \end{cases} \tag{5.9}$$

Let $u_i^{(2)} := \left\lfloor \frac{t}{q} \cdot f_i^{(2)} \right\rceil$. Then

$$D(c=2) = \left[u_0^{(2)} + u_1^{(2)} x + u_2^{(2)} x^2 + \cdots + u_{n-1}^{(2)} x^{n-1}\right]_t$$

As before, $u_i^{(2)}$ can have only t different possible values, and can be written as $u_i^{(2)} = \left\lceil -\frac{t}{2} \right\rceil + k_{i,2}, \quad$ with $k_{i,2} \in \{0, 1, \ldots, t-1\}$, and also $u_i^{(2)} = \left\lceil -\frac{t}{2} \right\rceil + k_{i,2} \Leftrightarrow -\frac{q}{2} + \frac{q}{t} k_{i,2} < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,2}+1)$. As before, since $-q/2 < f_i^{(2)} < q/2$ and t is odd, we have $\left\lceil -\frac{t}{2} \right\rceil \leq u_i^{(2)} \leq \left\lfloor \frac{t}{2} \right\rfloor$, and therefore we can simply write $D(c=2) = u_0^{(2)} + u_1^{(2)} x + u_2^{(2)} x^2 + \cdots + u_{n-1}^{(2)} x^{n-1} = \sum_{i=0}^{n-1} \left(\left\lceil -\frac{t}{2} \right\rceil + k_{i,2} \right) x^i$. So now, for each $0 \leq i \leq n-1$, we know $k_{i,1}, k_{i,2}$ such that

$$\begin{cases} -\frac{q}{2} + \frac{q}{t} k_{i,1} < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1}+1) \\ -\frac{q}{2} + \frac{q}{t} k_{i,2} < [2f_i]_q < -\frac{q}{2} + \frac{q}{t}(k_{i,2}+1) \end{cases}$$

There are 3 cases to distinguish, where $3 = 2^2 - 1$.

$(1/3)_{[c=2]}$. If $-\frac{q}{2} + \frac{q}{t}(k_{i,1}+1) \leq -\frac{q}{4} \wedge -\frac{q}{2} + \frac{q}{t} k_{i,1} \geq -\frac{q}{2}$, which says that $0 \leq k_{i,1} \leq \left\lfloor \frac{t}{4} - 1 \right\rfloor$, then we are sure that $f_i \in (-\frac{q}{2}, -\frac{q}{4})$. Therefore, by condition (5.9), we expect $f_i^{(2)} = [2f_i]_q = 2f_i + q$. Therefore, $-\frac{3q}{4} + \frac{q}{2t} k_{i,2} < f_i < -\frac{3q}{4} + \frac{q}{2t}(k_{i,2}+1)$

$(2/3)_{[c=2]}$. If $-\frac{q}{2} + \frac{q}{t}(k_{i,1}+1) \leq \frac{q}{4} \wedge -\frac{q}{2} + \frac{q}{t} k_{i,1} \geq -\frac{q}{4}$, which says that $\left\lceil \frac{t}{4} \right\rceil \leq k_{i,1} \leq \left\lfloor \frac{3t}{4} - 1 \right\rfloor$, then we are sure that $f_i \in (-\frac{q}{4}, \frac{q}{4})$. Therefore, by condition (5.9), we expect $f_i^{(2)} = [2f_i]_q = 2f_i$. Therefore, $-\frac{q}{4} + \frac{q}{2t} k_{i,2} < f_i < -\frac{q}{4} + \frac{q}{2t}(k_{i,2}+1)$

$(3/3)_{[c=2]}$. If $-\frac{q}{2} + \frac{q}{t}(k_{i,1}+1) \leq \frac{q}{2} \wedge -\frac{q}{2} + \frac{q}{t} k_{i,1} \geq \frac{q}{4}$, which says that $\left\lceil \frac{3t}{4} \right\rceil \leq k_{i,1} \leq t-1$, then we are sure that $f_i \in (\frac{q}{4}, \frac{q}{2})$. Therefore, by condition (5.9), we expect $f_i^{(2)} = [2f_i]_q = 2f_i - q$. Therefore, $\frac{q}{4} + \frac{q}{2t} k_{i,2} < f_i < \frac{q}{4} + \frac{q}{2t}(k_{i,2}+1)$

Now, we remark that there are values of $k_{i,1}$ for which is not clear to which of the previous cases we are falling in. For instance, if $k_{i,1}$ is such that $-\frac{q}{4} \in \left(-\frac{q}{2} + \frac{q}{t} k_{i,1}, -\frac{q}{2} + \frac{q}{t}(k_{i,1}+1) \right)$, then we are not sure whether we are in Case $(1/3)_{[c=2]}$ or in Case $(2/3)_{[c=2]}$. This uncertainty happens when $\nexists k_{i,1} \in [0, 1, \ldots, t-1]$ such that $-\frac{q}{2} + \frac{q}{t}k_{i,1} = -\frac{q}{4}$, i.e. such that $k_{i,1} = t/4$. So, if $\nexists k_{i,1} \in [0, 1, \ldots, t-1]$ such that $k_{i,1} = t/4$, i.e. if $4 \nmid t$, then $-\frac{q}{4} \in \left(-\frac{q}{2} + \frac{q}{t}\left\lfloor\frac{t}{4}\right\rfloor, -\frac{q}{2} + \frac{q}{t}\left(\left\lfloor\frac{t}{4}\right\rfloor + 1\right)\right)$. So, if $k_{i,1} = \left\lfloor\frac{t}{4}\right\rfloor$, with $\frac{t}{4} \notin \mathbb{N}$, we have that

$$f_i \in \left(-\frac{q}{2} + \frac{q}{t}\left\lfloor\frac{t}{4}\right\rfloor, -\frac{q}{2} + \frac{q}{t}\left(\left\lfloor\frac{t}{4}\right\rfloor + 1\right)\right) =: I$$

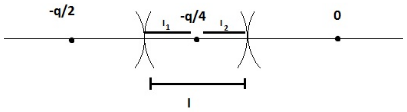
It is easy to see that

$$-\frac{q}{2} + \frac{q}{t}\left(\left\lfloor\frac{t}{4}\right\rfloor + 1\right) \leq 0, \forall 1 < t < q \tag{5.10}$$

There are two cases:

1/2: $f_i \in I_1 := I \cap (-q/2, -q/4)$. Then condition (5.9) implies that $f_i^{(2)} = [2f_i]_q \in (0, q/2)$

2/2: $f_i \in I_2 := I \cap (-q/4, 0)$. Then $f_i^{(2)} = [2f_i]_q \in (-q/2, 0)$



So, to sum up we have that if $k_{i,1} = \left\lfloor\frac{t}{4}\right\rfloor$, with $\frac{t}{4} \notin \mathbb{N}$, then

- if $f_i^{(2)} \in (0, q/2)$ then $f_i \in (-q/2, -q/4)$ and apply Case $(1/3)_{[c=2]}$
- if $f_i^{(2)} \in (-q/2, 0)$ then $f_i \in (-q/4, 0)$ and apply Case $(2/3)_{[c=2]}$

Similarly to what just discussed, if $k_{i,1}$ is such that

$$\frac{q}{4} \in \left(-\frac{q}{2} + \frac{q}{t}k_{i,1}, -\frac{q}{2} + \frac{q}{t}(k_{i,1} + 1)\right)$$

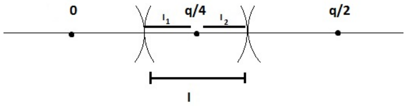
then we are not sure if we are in Case $(2/3)_{[c=2]}$ or in Case $(3/3)_{[c=2]}$ This uncertainty happens when $\nexists k_{i,1} \in [0, 1, \ldots, t-1]$ such that $-\frac{q}{2} + \frac{q}{t}k_{i,1} = \frac{q}{4}$, i.e. such that $k_{i,1} = 3t/4$. So, if $\nexists k_{i,1} \in [0, 1, \ldots, t-1]$ such that $k_{i,1} = 3t/4$, then $\frac{q}{4} \in \left(-\frac{q}{2} + \frac{q}{t}\left\lfloor\frac{3t}{4}\right\rfloor, -\frac{q}{2} + \frac{q}{t}\left(\left\lfloor\frac{3t}{4}\right\rfloor + 1\right)\right)$. So, if $k_{i,1} = \left\lfloor\frac{3t}{4}\right\rfloor$, with $\frac{3t}{4} \notin \mathbb{N}$, we have that $f_i \in \left(-\frac{q}{2} + \frac{q}{t}\left\lfloor\frac{3t}{4}\right\rfloor, -\frac{q}{2} + \frac{q}{t}\left(\left\lfloor\frac{3t}{4}\right\rfloor + 1\right)\right) =: I$. It is easy to see that

$$-\frac{q}{2} + \frac{q}{t}\left\lfloor\frac{3t}{4}\right\rfloor \geq 0, \forall t, q \tag{5.11}$$

There are two cases:

1/2: $f_i \in I_1 := I \cap (0, q/4)$. Then $f_i^{(2)} = [2f_i]_q \in (0, q/2)$

2/2: $f_i \in I_2 := I \cap (q/4, q/2)$. Then condition (5.9) implies that $f_i^{(2)} = [2f_i]_q \in (-q/2, 0)$



So, to sum up we have that if $k_{i,1} = \left\lfloor\frac{3t}{4}\right\rfloor$, with $\frac{3t}{4} \notin \mathbb{N}$, then

- if $f_i^{(2)} \in (0, q/2)$ then $f_i \in (-q/4, q/4)$ and apply Case $(2/3)_{[c=2]}$
- if $f_i^{(2)} \in (-q/2, 0)$ then $f_i \in (q/4, q/2)$ and apply Case $(3/3)_{[c=2]}$

We can write now all the 3 cases in a more complete way:

$(1/3)_{[c=2]}$. Suppose that

$$0 \leq k_{i,1} \leq \left\lfloor\frac{t}{4} - 1\right\rfloor \vee \left(k_{i,1} = \left\lfloor\frac{t}{4}\right\rfloor, \text{with } \frac{t}{4} \notin \mathbb{N} \wedge f_i^{(2)} \in (0, q/2)\right) \tag{K(1,1)}$$

Then

$$f_i \in \left(-\frac{q}{2}, -\frac{q}{4}\right), \qquad -\frac{3q}{4} + \frac{q}{2t}k_{i,2} < f_i < -\frac{3q}{4} + \frac{q}{2t}(k_{i,2}+1) \tag{F((1,1))}$$

$(2/3)_{[c=2]}$. Suppose that

$$\left\lceil \frac{t}{4} \right\rceil \leq k_{i,1} \leq \left\lfloor \frac{3t}{4} - 1 \right\rfloor \vee \left(k_{i,1} = \left\lfloor \frac{t}{4} \right\rfloor \wedge f_i^{(2)} \in (-q/2, 0)\right) \vee \\ \vee \left(k_{i,1} = \left\lfloor \frac{3t}{4} \right\rfloor \wedge f_i^{(2)} \in (0, q/2)\right) \tag{K(1,2)}$$

Then

$$f_i \in \left(-\frac{q}{4}, \frac{q}{4}\right), \qquad -\frac{q}{4} + \frac{q}{2t}k_{i,2} < f_i < -\frac{q}{4} + \frac{q}{2t}(k_{i,2}+1) \tag{F(1,2)}$$

$(3/3)_{[c=2]}$. Suppose that

$$\left\lceil \frac{3t}{4} \right\rceil \leq k_{i,1} \leq t-1 \vee \left(k_{i,1} = \left\lfloor \frac{3t}{4} \right\rfloor \wedge f_i^{(2)} \in (-q/2, 0)\right) \tag{K(1,3)}$$

Then

$$f_i \in \left(\frac{q}{4}, \frac{q}{2}\right), \qquad \frac{q}{4} + \frac{q}{2t}k_{i,2} < f_i < \frac{q}{4} + \frac{q}{2t}(k_{i,2}+1) \tag{F(1,3)}$$

In all cases, we end up by knowing f_i with an error up to $\frac{q}{2t}$.

Step 3: select $c(x) = 4$

Select now "ciphertext" $c(x) = 4 = 4 + 0x + 0x^2 + \cdots + 0x^{n-1}$. Decryption oracle computes and return the polynomial

$$
\begin{aligned}
D(c = 4) &= \left[\left\lfloor \frac{t}{q} \cdot [f \cdot 4]_q \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t}{q} \cdot \left([4f_0]_q + [4f_1]_q x + [4f_2]_q x^2 + \cdots + [4f_{n-1}]_q x^{n-1} \right) \right\rceil \right]_t \\
&= \left[\left\lfloor \frac{t}{q} f_0^{(3)} \right\rceil + \left\lfloor \frac{t}{q} f_1^{(3)} \right\rceil x + \cdots + \left\lfloor \frac{t}{q} f_{n-1}^{(3)} \right\rceil x^{n-1} \right]_t
\end{aligned}
$$

where we have put $f_i^{(3)} := [4f_i]_q$, for every $0 \leq i \leq n-1$; of course we have $-\frac{q}{2} < f_i^{(3)} < \frac{q}{2}$.

Now,

- if $-q/8 < f_i < q/8$, then

  $$-\frac{q}{2} < 4f_i < \frac{q}{2}$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i$$

- if $-q/4 < f_i < -q/8$, then

  $$-q < 4f_i < -\frac{q}{2}$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i + q$$

- if $q/8 < f_i < q/4$, then

  $$\frac{q}{2} < 4f_i < q$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i - q$$

- if $-3q/8 < f_i < -q/4$, then

  $$-\frac{3q}{2} < 4f_i < -q$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i + q$$

- if $q/4 < f_i < 3q/8$, then

  $$q < 4f_i < \frac{3q}{2}$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i - q$$

- if $-q/2 < f_i < -3q/8$, then

  $$-2q < 4f_i < -\frac{3q}{2}$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i + 2q$$

- if $3q/8 < f_i < q/2$, then

  $$\frac{3q}{2} < 4f_i < 2q$$

  and therefore

  $$f_i^{(3)} = [4f_i]_q = 4f_i - 2q$$

So we have

$$f_i^{(3)} = [4f_i]_q = \begin{cases} 4f_i & \text{if } -\frac{q}{8} < f_i < \frac{q}{8} \\ 4f_i + q & \text{if } -\frac{q}{4} < f_i < -\frac{q}{8}, \text{ and in this case } 0 < f_i^{(3)} < \frac{q}{2} \\ 4f_i - q & \text{if } \frac{q}{8} < f_i < \frac{q}{4}, \text{ and in this case } -\frac{q}{2} < f_i^{(3)} < 0 \\ 4f_i + q & \text{if } -\frac{3q}{8} < f_i < -\frac{q}{4}, \text{ and in this case } -\frac{q}{2} < f_i^{(3)} < 0 \\ 4f_i - q & \text{if } \frac{q}{4} < f_i < \frac{3q}{8}, \text{ and in this case } 0 < f_i^{(3)} < \frac{q}{2} \\ 4f_i + 2q & \text{if } -\frac{q}{2} < f_i < -\frac{3q}{8}, \text{ and in this case } 0 < f_i^{(3)} < \frac{q}{2} \\ 4f_i - 2q & \text{if } \frac{3q}{8} < f_i < \frac{q}{2}, \text{ and in this case } -\frac{q}{2} < f_i^{(3)} < 0 \end{cases} \tag{5.12}$$

Let $u_i^{(3)} := \left\lfloor \frac{t}{q} \cdot f_i^{(3)} \right\rceil$. We have

$$D(c = 4) = \left[u_0^{(3)} + u_1^{(3)}x + u_2^{(3)}x^2 + \cdots + u_{n-1}^{(3)}x^{n-1}\right]_t$$

As before, $u_i^{(3)}$ can have only t different possible values, and can be written as

$$u_i^{(3)} = \left\lceil -\frac{t}{2} \right\rceil + k_{i,3}, \quad \text{with } k_{i,3} \in \{0, 1, \ldots, t-1\}$$

As before, we have

$$u_i^{(3)} = \left\lceil -\frac{t}{2} \right\rceil + k_{i,3} \Leftrightarrow -\frac{q}{2} + \frac{q}{t}k_{i,3} < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,3} + 1)$$

As before, since $-q/2 < f_i^{(3)} < q/2$ and t is odd, we have

$$-\frac{t}{2} < \left\lceil -\frac{t}{2} \right\rceil \leq u_i^{(3)} \leq \left\lfloor \frac{t}{2} \right\rfloor$$

and therefore we can simply write

$$D(c=4) = u_0^{(3)} + u_1^{(3)}x + u_2^{(3)}x^2 + \cdots + u_{n-1}^{(3)}x^{n-1} = \sum_{i=0}^{n-1}\left(\left\lceil -\frac{t}{2}\right\rceil + k_{i,3}\right)$$

and therefore we learn integers $k_{i,3}$, for $0 \leq i \leq n-1$. Now, for each $0 \leq i \leq n-1$, we know $k_{i,1}, k_{i,2}, k_{i,3}$ such that

$$\begin{cases} -\frac{q}{2} + \frac{q}{t}k_{i,1} < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1}+1) \\ -\frac{q}{2} + \frac{q}{t}k_{i,2} < [2f_i]_q < -\frac{q}{2} + \frac{q}{t}(k_{i,2}+1) \\ -\frac{q}{2} + \frac{q}{t}k_{i,3} < [4f_i]_q < -\frac{q}{2} + \frac{q}{t}(k_{i,3}+1) \end{cases}$$

There are $7 = 2^3 - 1$ cases to distinguish.

$(1/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(1/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies condition (K(1,1)) and f_i satisfies condition (F((1,1)):

$$f_i \in \left(-\frac{q}{2}, -\frac{q}{4}\right), \qquad -\frac{3q}{4} + \frac{q}{2t}k_{i,2} < f_i < -\frac{3q}{4} + \frac{q}{2t}(k_{i,2}+1)$$

If

$$-\frac{3q}{4} + \frac{q}{2t}(k_{i,2}+1) \leq -\frac{3q}{8} \wedge -\frac{3q}{4} + \frac{q}{2t}k_{i,2} \geq -\frac{q}{2}$$

which says that

$$\left\lceil \frac{t}{2} \right\rceil \leq k_{i,2} \leq \left\lfloor \frac{3t}{4} - 1 \right\rfloor$$

then we are sure that $f_i \in (-\frac{q}{2}, -\frac{3q}{8})$. Therefore, by condition (5.12), we expect $f_i^{(3)} = [4f_i]_q = 4f_i + 2q$. Therefore,

$$-\frac{q}{2} + \frac{q}{t}k_{i,3} < 4f_i + 2q < -\frac{q}{2} + \frac{q}{t}(k_{i,3}+1)$$

$$-\frac{5q}{8} + \frac{q}{4t}k_{i,3} < f_i < -\frac{5q}{8} + \frac{q}{4t}(k_{i,3}+1)$$

$(2/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(1/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies again condition (K(1,1)) and f_i satisfies condition (F((1,1)). If

$$-\frac{3q}{4}+\frac{q}{2t}(k_{i,2}+1)\leq -\frac{q}{4}\wedge -\frac{3q}{4}+\frac{q}{2t}k_{i,2}\geq -\frac{3q}{8}$$

which says that

$$\left\lceil\frac{3t}{4}\right\rceil\leq k_{i,2}\leq t-1$$

then we are sure that $f_i\in(-\frac{3q}{8},-\frac{q}{4})$. Therefore, by condition (5.12), we expect $f_i^{(3)}=[4f_i]_q=4f_i+q$. Therefore,

$$-\frac{q}{2}+\frac{q}{t}k_{i,3}<4f_i+q<-\frac{q}{2}+\frac{q}{t}(k_{i,3}+1)$$

$$-\frac{3q}{8}+\frac{q}{4t}k_{i,3}<f_i<-\frac{3q}{8}+\frac{q}{4t}(k_{i,3}+1)$$

$(3/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(2/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies condition (K(1,2)) and f_i satisfies condition (F(1,2)):

$$f_i\in\left(-\frac{q}{4},\frac{q}{4}\right),\qquad -\frac{q}{4}+\frac{q}{2t}k_{i,2}<f_i<-\frac{q}{4}+\frac{q}{2t}(k_{i,2}+1)$$

If

$$-\frac{q}{4}+\frac{q}{2t}(k_{i,2}+1)\leq -\frac{q}{8}\wedge -\frac{q}{4}+\frac{q}{2t}k_{i,2}\geq -\frac{q}{4}$$

which says that

$$0\leq k_{i,2}\leq\left\lfloor\frac{t}{4}-1\right\rfloor$$

then we are sure that $f_i\in(-\frac{q}{4},-\frac{q}{8})$. Therefore, by condition (5.12), we expect $f_i^{(3)}=[4f_i]_q=4f_i+q$. Therefore,

$$-\frac{q}{2}+\frac{q}{t}k_{i,3}<4f_i+q<-\frac{q}{2}+\frac{q}{t}(k_{i,3}+1)$$

$$-\frac{3q}{8}+\frac{q}{4t}k_{i,3}<f_i<-\frac{3q}{8}+\frac{q}{4t}(k_{i,3}+1)$$

$(4/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(2/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies again condition (K(1,2)) and f_i satisfies condition (F(1,2)). If

$$-\frac{q}{4}+\frac{q}{2t}(k_{i,2}+1)\leq\frac{q}{8}\wedge-\frac{q}{4}+\frac{q}{2t}k_{i,2}\geq-\frac{q}{8}$$

which says that

$$\left\lceil\frac{t}{4}\right\rceil\leq k_{i,2}\leq\left\lfloor\frac{3t}{4}-1\right\rfloor$$

then we are sure that $f_i\in(-\frac{q}{8},\frac{q}{8})$. Therefore, by condition (5.12), we expect $f_i^{(3)}=[4f_i]_q=4f_i$. Therefore,

$$-\frac{q}{2}+\frac{q}{t}k_{i,3}<4f_i<-\frac{q}{2}+\frac{q}{t}(k_{i,3}+1)$$

$$-\frac{q}{8}+\frac{q}{4t}k_{i,3}<f_i<-\frac{q}{8}+\frac{q}{4t}(k_{i,3}+1)$$

$(5/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(2/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies again condition (K(1,2)) and f_i satisfies condition (F(1,2)). If

$$-\frac{q}{4}+\frac{q}{2t}(k_{i,2}+1)\leq\frac{q}{4}\wedge-\frac{q}{4}+\frac{q}{2t}k_{i,2}\geq\frac{q}{8}$$

which says that

$$\left\lceil\frac{3t}{4}\right\rceil\leq k_{i,2}\leq t-1$$

then we are sure that $f_i\in(\frac{q}{8},\frac{q}{4})$. Therefore, by condition (5.12), we expect $f_i^{(3)}=[4f_i]_q=4f_i-q$. Therefore,

$$-\frac{q}{2}+\frac{q}{t}k_{i,3}<4f_i-q<-\frac{q}{2}+\frac{q}{t}(k_{i,3}+1)$$

$$\frac{q}{8} + \frac{q}{4t}k_{i,3} < f_i < \frac{q}{8} + \frac{q}{4t}(k_{i,3} + 1)$$

$(6/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(3/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies condition (K(1,3)) and f_i satisfies condition (F(1,3)):

$$f_i \in \left(\frac{q}{4}, \frac{q}{2}\right), \qquad \frac{q}{4} + \frac{q}{2t}k_{i,2} < f_i < \frac{q}{4} + \frac{q}{2t}(k_{i,2} + 1)$$

If

$$\frac{q}{4} + \frac{q}{2t}(k_{i,2} + 1) \leq \frac{3q}{8} \wedge \frac{q}{4} + \frac{q}{2t}k_{i,2} \geq \frac{q}{4}$$

which says that

$$0 \leq k_{i,2} \leq \left\lfloor \frac{t}{4} - 1 \right\rfloor$$

then we are sure that $f_i \in (\frac{q}{4}, \frac{3q}{8})$. Therefore, by condition (5.12), we expect $f_i^{(3)} = [4f_i]_q = 4f_i - q$. Therefore,

$$-\frac{q}{2} + \frac{q}{t}k_{i,3} < 4f_i - q < -\frac{q}{2} + \frac{q}{t}(k_{i,3} + 1)$$

$$\frac{q}{8} + \frac{q}{4t}k_{i,3} < f_i < \frac{q}{8} + \frac{q}{4t}(k_{i,3} + 1)$$

$(7/7)_{[c=4]}$. Suppose that at Step 2, we were in case $(3/3)_{[c=2]}$, i.e. $k_{i,1}$ satisfies again condition (K(1,3)) and f_i satisfies condition (F(1,3)). If

$$\frac{q}{4} + \frac{q}{2t}(k_{i,2} + 1) \leq \frac{q}{2} \wedge \frac{q}{4} + \frac{q}{2t}k_{i,2} \geq \frac{3q}{8}$$

which says that

$$\left\lceil \frac{t}{4} \right\rceil \leq k_{i,2} \leq \left\lfloor \frac{t}{2} - 1 \right\rfloor$$

then we are sure that $f_i \in (\frac{3q}{8}, \frac{q}{2})$. Therefore, by condition (5.12), we expect $f_i^{(3)} = [4f_i]_q = 4f_i - 2q$. Therefore,

$$-\frac{q}{2} + \frac{q}{t}k_{i,3} < 4f_i - 2q < -\frac{q}{2} + \frac{q}{t}(k_{i,3}+1)$$

$$\frac{3q}{8} + \frac{q}{4t}k_{i,3} < f_i < \frac{3q}{8} + \frac{q}{4t}(k_{i,3}+1)$$

We remark that, similarly as what we discussed in Step 2, there are values of $k_{i,2}$ for which is not clear immediately to which of the previous cases we are falling in. For instance, if in Step 2 we were in case $(1/3)_{[c=2]}$, and if $k_{i,2}$ is such that

$$-\frac{3q}{8} \in \left(-\frac{3q}{4} + \frac{q}{2t}k_{i,2}, -\frac{3q}{4} + \frac{q}{2t}(k_{i,2}+1)\right)$$

then we are not sure if we are in Case $(1/7)_{[c=4]}$ or in Case $(2/7)_{[c=4]}$. This uncertainty happens when $\nexists k_{i,2} \in [0, 1, \ldots, t-1]$ such that

$$-\frac{3q}{4} + \frac{q}{2t}k_{i,2} = -\frac{3q}{8}$$

i.e. such that $k_{i,2} = 3t/4$. So, if $\nexists k_{i,2} \in [0, 1, \ldots, t-1]$ such that $k_{i,2} = 3t/4$, i.e. if $4 \nmid 3t$, i.e. if $4 \nmid t$, then

$$-\frac{3q}{8} \in \left(-\frac{3q}{4} + \frac{q}{2t}\left\lfloor\frac{3t}{4}\right\rfloor, -\frac{3q}{4} + \frac{q}{2t}\left(\left\lfloor\frac{3t}{4}\right\rfloor + 1\right)\right)$$

So, if $k_{i,2} = \left\lfloor\frac{3t}{4}\right\rfloor$, with $\frac{3t}{4} \notin \mathbb{N}$, we have the following.

$$f_i \in \left(-\frac{3q}{4} + \frac{q}{2t}\left\lfloor\frac{3t}{4}\right\rfloor, -\frac{3q}{4} + \frac{q}{2t}\left(\left\lfloor\frac{3t}{4}\right\rfloor + 1\right)\right) =: I$$

There are two cases:

1/2: $f_i \in I_1 := I \cap (-q/2, -3q/8)$

Then condition (5.12) implies that $f_i^{(3)} = [4f_i]_q \in (0, q/2)$

2/2: $f_i \in I_2 := I \cap (-3q/8, -q/4)$

Then condition (5.12) implies that $f_i^{(3)} = [4f_i]_q \in (-q/2, 0)$

So, to sum up we have that if $k_{i,2} = \left\lfloor \frac{3t}{4} \right\rfloor$, with $\frac{3t}{4} \notin \mathbb{N}$, then

- if $f_i^{(3)} \in (0, q/2)$ then $f_i \in (-q/2, -3q/8)$ and apply Case $(1/7)_{[c=4]}$
- if $f_i^{(3)} \in (-q/2, 0)$ then $f_i \in (-3q/8, -q/4)$ and apply Case $(2/7)_{[c=4]}$

Similarly to what just discussed, if in Step 2 we were in case $(2/3)_{[c=2]}$, and if $k_{i,2}$ is such that

$$-\frac{q}{8} \in \left(-\frac{q}{4} + \frac{q}{2t}k_{i,2}, -\frac{q}{4} + \frac{q}{2t}(k_{i,2}+1)\right)$$

then we are not sure if we are in Case $(3/7)_{[c=4]}$ or in Case $(4/7)_{[c=4]}$. Again, reasoning in similar way as before, if $k_{i,2} = \left\lfloor \frac{t}{4} \right\rfloor$, with $\frac{t}{4} \notin \mathbb{N}$, we have the following.

$$f_i \in \left(-\frac{q}{4} + \frac{q}{2t}\left\lfloor \frac{t}{4} \right\rfloor, -\frac{q}{4} + \frac{q}{2t}\left(\left\lfloor \frac{t}{4} \right\rfloor + 1\right)\right) =: I$$

It is easy to see that

$$-\frac{q}{4} + \frac{q}{2t}\left(\left\lfloor \frac{t}{4} \right\rfloor + 1\right) \leq 0, \forall t, q \tag{5.13}$$

There are two cases:

1/2: $f_i \in I_1 := I \cap (-q/4, -q/8)$

Then condition (5.12) implies that $f_i^{(3)} = [4f_i]_q \in (0, q/2)$

2/2: $f_i \in I_2 := I \cap (-q/8, 0)$

Then $f_i^{(3)} = [4f_i]_q \in (-q/2, 0)$

So, to sum up we have that if $k_{i,2} = \left\lfloor \frac{t}{4} \right\rfloor$, with $\frac{t}{4} \notin \mathbb{N}$, then

- if $f_i^{(3)} \in (0, q/2)$ then $f_i \in (-q/4, -q/8)$ and apply Case $(3/7)_{[c=4]}$
- if $f_i^{(3)} \in (-q/2, 0)$ then $f_i \in (-q/8, 0)$ and apply Case $(4/7)_{[c=4]}$

Similarly, if in Step 2 we were in case $(2/3)_{[c=2]}$, and if $k_{i,2}$ is such that

$$\frac{q}{8} \in \left(-\frac{q}{4} + \frac{q}{2t}k_{i,2}, -\frac{q}{4} + \frac{q}{2t}(k_{i,2}+1)\right)$$

then we are not sure if we are in Case $(4/7)_{[c=4]}$ or in Case $(5/7)_{[c=4]}$. Again, reasoning in similar way as before, if $k_{i,2} = \left\lfloor \frac{3t}{4} \right\rfloor$, with $\frac{3t}{4} \notin \mathbb{N}$, we have the following.

$$f_i \in \left(-\frac{q}{4} + \frac{q}{2t}\left\lfloor \frac{3t}{4} \right\rfloor, -\frac{q}{4} + \frac{q}{2t}\left(\left\lfloor \frac{3t}{4} \right\rfloor + 1\right)\right) =: I$$

It is easy to see that

$$-\frac{q}{4} + \frac{q}{2t}\left\lfloor \frac{3t}{4} \right\rfloor \geq 0, \forall t, q \tag{5.14}$$

There are two cases:

1/2: $f_i \in I_1 := I \cap (0, q/8)$

Then $f_i^{(3)} = [4f_i]_q \in (0, q/2)$

2/2: $f_i \in I_2 := I \cap (q/8, q/4)$

Then condition (5.12) implies that $f_i^{(3)} = [4f_i]_q \in (-q/2, 0)$

So, to sum up we have that if $k_{i,2} = \left\lfloor \frac{3t}{4} \right\rfloor$, with $\frac{3t}{4} \notin \mathbb{N}$, then

- if $f_i^{(3)} \in (0, q/2)$ then $f_i \in (0, q/8)$ and apply Case $(4/7)_{[c=4]}$
- if $f_i^{(3)} \in (-q/2, 0)$ then $f_i \in (q/8, q/4)$ and apply Case $(5/7)_{[c=4]}$

Similarly, if in Step 2 we were in case $(3/3)_{[c=2]}$, and if $k_{i,2}$ is such that

$$\frac{3q}{8} \in \left(\frac{q}{4} + \frac{q}{2t}k_{i,2}, \frac{q}{4} + \frac{q}{2t}(k_{i,2}+1)\right)$$

then we are not sure if we are in Case $(6/7)_{[c=4]}$ or in Case $(7/7)_{[c=4]}$. Again, reasoning in similar way as before, if $k_{i,2} = \left\lfloor \frac{t}{4} \right\rfloor$, with $\frac{t}{4} \notin \mathbb{N}$, we have the following.

$$f_i \in \left(\frac{q}{4} + \frac{q}{2t}\left\lfloor \frac{t}{4} \right\rfloor, \frac{q}{4} + \frac{q}{2t}\left(\left\lfloor \frac{t}{4} \right\rfloor + 1\right)\right) =: I$$

There are two cases:

1/2: $f_i \in I_1 := I \cap (q/4, 3q/8)$
Then condition (5.12) implies that $f_i^{(3)} = [4f_i]_q \in (0, q/2)$

2/2: $f_i \in I_2 := I \cap (3q/8, q/2)$
Then condition (5.12) implies that $f_i^{(3)} = [4f_i]_q \in (-q/2, 0)$

So, to sum up we have that if $k_{i,2} = \left\lfloor \frac{t}{4} \right\rfloor$, with $\frac{t}{4} \notin \mathbb{N}$, then

- if $f_i^{(3)} \in (0, q/2)$ then $f_i \in (q/4, 3q/8)$ and apply Case $(6/7)_{[c=4]}$
- if $f_i^{(3)} \in (-q/2, 0)$ then $f_i \in (3q/8, q/2)$ and apply Case $(7/7)_{[c=4]}$

$(1/7)_{[c=4]}$. Suppose that

$$\text{Condition (K(1,1)) holds } \textbf{and}$$
$$\left(\left\lceil \frac{t}{2} \right\rceil \le k_{i,2} \le \left\lfloor \frac{3t}{4} - 1 \right\rfloor \vee \left(k_{i,2} = \left\lfloor \frac{3t}{4} \right\rfloor \wedge f_i^{(3)} \in \left(0, \frac{q}{2}\right)\right)\right) \tag{K(2,1)}$$

Then

$$f_i \in (-\frac{q}{2}, -\frac{3q}{8}), \qquad -\frac{5q}{8} + \frac{q}{4t}k_{i,3} < f_i < -\frac{5q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,1)}$$

$(2/7)_{[c=4]}$. Suppose that

$$\text{Condition (K(1,1)) holds } \textbf{and} \\ \left(\left\lceil\frac{3t}{4}\right\rceil \leq k_{i,2} \leq t-1 \vee \left(k_{i,2} = \left\lfloor\frac{3t}{4}\right\rfloor \wedge f_i^{(3)} \in \left(-\frac{q}{2}, 0\right)\right)\right) \tag{K(2,2)}$$

Then

$$f_i \in (-\frac{3q}{8}, -\frac{q}{4}), \qquad -\frac{3q}{8} + \frac{q}{4t}k_{i,3} < f_i < -\frac{3q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,2)}$$

$(3/7)_{[c=4]}$. Suppose that

$$\text{Condition (K(1,2)) holds } \textbf{and} \\ \left(0 \leq k_{i,2} \leq \left\lfloor\frac{t}{4} - 1\right\rfloor \vee \left(k_{i,2} = \left\lfloor\frac{t}{4}\right\rfloor \wedge f_i^{(3)} \in \left(0, \frac{q}{2}\right)\right)\right) \tag{K(2,3)}$$

Then

$$f_i \in (-\frac{q}{4}, -\frac{q}{8}), \qquad -\frac{3q}{8} + \frac{q}{4t}k_{i,3} < f_i < -\frac{3q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,3)}$$

$(4/7)_{[c=4]}$. Suppose that

$$\text{Condition (K(1,2)) holds } \textbf{and} \\ \left[\left\lceil\frac{t}{4}\right\rceil \leq k_{i,2} \leq \left\lfloor\frac{3t}{4} - 1\right\rfloor \vee \left(k_{i,2} = \left\lfloor\frac{t}{4}\right\rfloor \wedge f_i^{(3)} \in \left(-\frac{q}{2}, 0\right)\right) \vee \right. \\ \left. \vee \left(k_{i,2} = \left\lfloor\frac{3t}{4}\right\rfloor \wedge f_i^{(3)} \in \left(0, \frac{q}{2}\right)\right)\right] \tag{K(2,4)}$$

Then

$$f_i \in (-\frac{q}{8}, \frac{q}{8}), \qquad -\frac{q}{8} + \frac{q}{4t}k_{i,3} < f_i < -\frac{q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,4)}$$

$(5/7)_{[c=4]}$. Suppose that

$$\text{Condition (K(1,2)) holds } \textbf{and} \\ \left(\left\lceil\frac{3t}{4}\right\rceil \leq k_{i,2} \leq t-1 \vee \left(k_{i,2} = \left\lfloor\frac{3t}{4}\right\rfloor \wedge f_i^{(3)} \in \left(-\frac{q}{2}, 0\right)\right)\right) \tag{K(2,5)}$$

Then

$$f_i \in (\frac{q}{8}, \frac{q}{4}), \qquad \frac{q}{8} + \frac{q}{4t} k_{i,3} < f_i < \frac{q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,5)}$$

$(6/7)_{[c=4]}$. Suppose that

$$\begin{gathered}\text{Condition (K(1,3)) holds } \textbf{and} \\ \left(0 \leq k_{i,2} \leq \left\lfloor \frac{t}{4} - 1 \right\rfloor \vee \left(k_{i,2} = \left\lfloor \frac{t}{4} \right\rfloor \wedge f_i^{(3)} \in \left(0, \frac{q}{2}\right)\right)\right)\end{gathered} \tag{K(2,6)}$$

Then

$$f_i \in (\frac{q}{4}, \frac{3q}{8}), \qquad \frac{q}{8} + \frac{q}{4t} k_{i,3} < f_i < \frac{q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,6)}$$

$(7/7)_{[c=4]}$. Suppose that

$$\begin{gathered}\text{Condition (K(1,3)) holds } \textbf{and} \\ \left(\left\lceil \frac{t}{4} \right\rceil \leq k_{i,2} \leq \left\lfloor \frac{t}{2} - 1 \right\rfloor \vee \left(k_{i,2} = \left\lfloor \frac{t}{4} \right\rfloor \wedge f_i^{(3)} \in \left(-\frac{q}{2}, 0\right)\right)\right)\end{gathered} \tag{K(2,7)}$$

Then

$$f_i \in (\frac{3q}{8}, \frac{q}{2}), \qquad \frac{3q}{8} + \frac{q}{4t} k_{i,3} < f_i < \frac{3q}{8} + \frac{q}{4t}(k_{i,3}+1) \tag{F(2,7)}$$

In all cases, we end up by knowing f_i with an error up to $\frac{q}{4t}$.

Generalization and complexity

At each step, we keep submitting “ciphertexts" $c(x) := 2^h$, for increasing values $h = 0, 1, 2, \ldots$, i.e. at step $h+1$ we submit ciphertext $c(x) = 2^h$. Suppose we are at step $h+1$. Then we submit to the decryption oracle the 'ciphertext' $c(x) = 2^h$, and the decryption oracle will return us a polynomial

$$\begin{aligned} D(c = 2^h) &= u_0^{(h+1)} + u_1^{(h+1)}x + \cdots + u_{n-1}^{(h+1)}x^{n-1} = \sum_{i=0}^{n-1} u_i^{(h+1)}x^i \\ &= \sum_{i=0}^{n-1}\left(\left\lceil -\frac{t}{2}\right\rceil + k_{i,h+1}\right) \in R_t \end{aligned}$$

from which we learn values $k_{i,h+1}$ for $1 \leq i \leq n-1$. So, at this point, we know $k_{i,j}$, for $0 \leq i \leq n-1$ and $1 \leq j \leq h+1$. These values allow us to distinguish between $m_h := 2^{h+1} - 1$ cases: for each $0 \leq i \leq n-1$, we know that integer f_i belongs to one of the cases:

$(a/2^{h+1} - 1)_{[c=2^h]}$. Suppose that

$$\text{[Condition (C}(h, a, 1)\text{) holds]} \ \wedge \text{[Condition (C}(h, a, 2)\text{) holds]} \qquad \text{(K(h,a))}$$

Then

$$f_i \in (x_{a,h}, y_{a,h}), \qquad \Delta_{h,a} + \frac{q}{2^h t}k_{i,h+1} < f_i < \Delta_{h,a} + \frac{q}{2^h t}(k_{i,h+1} + 1) \qquad \text{(F(h,a))}$$

where $a \in \{1, 2, \ldots, 2^{h+1} - 1\}$. Since

$$\Delta_{h,a} + \frac{q}{2^h t}(k_{i,h+1} + 1) - \left(\Delta_{h,a} + \frac{q}{2^h t}k_{i,h+1}\right) = \frac{q}{2^h t},$$

this allows us to recover, for each $0 \leq i \leq n-1$, the integer f_i with an error up to $\frac{q}{2^h t}$. Therefore, we keep submitting 'ciphertexts' $c(x) = 2^h$ for increasing values $h = 0, 1, 2, \ldots$ until h is such that $\frac{q}{2^h t} < 1$, i.e. $h \geq \lceil \log_2(q/t) \rceil$. So, we have to repeat our attack, submitting ciphertexts $c(x) = 1 = 2^0, 2^1, 2^2, 2^3, \ldots, 2^H$, where $H := \lceil \log_2(q/t) \rceil$. Se we repeat our attack $H+1$ times. Now, the secret key is $f(x) = f_0 + f_1x + \cdots + f_{n-1}x^{n-1}$, where

$f_i \in (-q/2, q/2]$, $\forall 0 \leq i \leq n-1$. So f_i can have q different values. The decryption oracle reveals a polynomial $m(x) = m_0 + m_1 x + \cdots + m_{n-1}x^{n-1}$, where $m_i \in (-t/2, t/2]$, $\forall 0 \leq i \leq n-1$. So m_i can have t different values. Each f_i can be described with at most $\lfloor \log_2(q-1) \rfloor + 1$ bits. So $f(x)$ can be described with $n \cdot (\lfloor \log_2(q-1) \rfloor + 1)$. Oracle decryption reveals $n \cdot (\lfloor \log_2(t-1) \rfloor + 1)$ bits. So the minimum number of oracle queries to determine $f(x)$ is given by $\frac{n \cdot (\lfloor \log_2(q-1) \rfloor + 1)}{n \cdot (\lfloor \log_2(t-1) \rfloor + 1)}$. In order to finish our attack for t odd, we need to give complete description of $\Delta_{h,a}$, Condition $\mathrm{C}(h,a,1)$ and Condition $\mathrm{C}(h,a,2)$, for each $0 \leq h \leq \lceil \log_2(q/t) \rceil = H$ and for each $1 \leq a \leq 2^{h+1} - 1$. Fix $0 \leq h \leq \lceil \log_2(q/t) \rceil$. For a given $1 \leq a \leq 2^{h+1} - 1$ put

$$\delta_{h,a} := \begin{cases} 2^{h-1} & \text{if } a = 2^h \\ \left\lfloor \frac{a}{2} \right\rfloor & \text{if } 1 \leq a < 2^h \\ \left\lceil \frac{a}{2} \right\rceil & \text{if } 2^h < a \leq 2^{h+1} - 1 \end{cases} \quad , \quad \Delta_{h,a} := -\left(\frac{1}{2} + \frac{1}{2^{h+1}} - \frac{\delta_{h,a}}{2^h} \right) \cdot q$$

Also, put

$$\eta(h,a) := \begin{cases} \left\lceil \frac{a}{2} \right\rceil & \text{if } 1 \leq a \leq 2^h \\ \left\lfloor \frac{a}{2} \right\rfloor & \text{if } 2^h < a \leq 2^{h+1} - 1 \end{cases}$$

Then

$$\text{Condition } (\mathrm{C}(h,a,1)) = \text{Condition } (\mathrm{K}(h-1, \eta(h,a)))$$

Remark that, if $h = 0$ or $h = 1$, then Condition $(\mathrm{C}(h,a,1)) = \emptyset$ i.e., we don't put any condition at all, vacuous condition.

For Condition $\mathrm{C}(h,a,2)$, remark that if $h = 0$ then Condition $(\mathrm{C}(0,a,2)) = \emptyset$ i.e., we don't put any condition at all, vacuous condition. One can see that, at step $h+1$, condition $\mathrm{C}(h,a,2)$ is only one among the following 5:

1. $V_{3,h} := U_{2,1} = U_{1,1} \wedge (r \text{ is even}) = U_{3,1} \wedge (r \text{ is odd})$:

$$0 \leq k_{i,h} \leq \left\lfloor \frac{t}{4} - 1 \right\rfloor \vee \left(k_{i,h} = \left\lfloor \frac{t}{4} \right\rfloor \wedge f_i^{(h+1)} \in \left(0, \frac{q}{2}\right) \right) \tag{$V_{3,h}$}$$

2. $V_{5,h} := U_{2,2}$:

$$\begin{aligned} \left\lceil \frac{t}{4} \right\rceil \leq k_{i,h} \leq \left\lfloor \frac{3t}{4} - 1 \right\rfloor \vee \left(k_{i,h} = \left\lfloor \frac{t}{4} \right\rfloor \wedge f_i^{(h+1)} \in \left(-\frac{q}{2}, 0\right) \right) \vee \\ \vee \left(k_{i,h} = \left\lfloor \frac{3t}{4} \right\rfloor \wedge f_i^{(h+1)} \in \left(0, \frac{q}{2}\right) \right) \end{aligned} \tag{$V_{5,h}$}$$

3. $V_{2,h} := U_{2,3} = U_{1,2} \wedge (r \text{ is odd}) = U_{3,2} \wedge (r \text{ is even})$:

$$\left\lceil \frac{3t}{4} \right\rceil \leq k_{i,h} \leq t - 1 \vee \left(k_{i,h} = \left\lfloor \frac{3t}{4} \right\rfloor \wedge f_i^{(h+1)} \in \left(-\frac{q}{2}, 0\right) \right) \tag{$V_{2,h}$}$$

4. $V_{1,h} := U_{1,1} \wedge (r \text{ is odd}) = U_{3,1} \wedge (r \text{ is even})$:

$$\left\lceil \frac{t}{2} \right\rceil \leq k_{i,h} \leq \left\lfloor \frac{3t}{4} - 1 \right\rfloor \vee \left(k_{i,h} = \left\lfloor \frac{3t}{4} \right\rfloor \wedge f_i^{(h+1)} \in \left(0, \frac{q}{2}\right) \right) \tag{$V_{1,h}$}$$

5. $V_{0,h} := U_{1,2} \wedge (r \text{ is even}) = U_{3,2} \wedge (r \text{ is odd})$:

$$\left\lceil \frac{t}{4} \right\rceil \leq k_{i,h} \leq \left\lfloor \frac{t}{2} - 1 \right\rfloor \vee \left(k_{i,h} = \left\lfloor \frac{t}{4} \right\rfloor \wedge f_i^{(h+1)} \in \left(-\frac{q}{2}, 0\right) \right) \tag{$V_{0,h}$}$$

So, suppose we are in case $(a/2^{h+1} - 1)_{[c=2^h]}$. Then we see that we have

Therefore, we have

$$\mathrm{C}(h,a,2)=\begin{cases} V_{1,h} & \text{if } 1\le a\le 2^h-2\wedge a\equiv 1 \bmod 4 \textbf{ or}\\ & \quad 2^h+2\le a\le 2^{h+1}-1\wedge a\equiv 0 \bmod 4\\ V_{2,h} & \text{if } 1\le a\le 2^h-2\wedge a\equiv 2 \bmod 4 \textbf{ or}\\ & \quad 2^h+2\le a\le 2^{h+1}-1\wedge a\equiv 1 \bmod 4 \textbf{ or}\\ & \quad a=2^h+1\\ V_{3,h} & \text{if } 1\le a\le 2^h-2\wedge a\equiv 3 \bmod 4 \textbf{ or}\\ & \quad 2^h+2\le a\le 2^{h+1}-1\wedge a\equiv 2 \bmod 4 \textbf{ or}\\ & \quad a=2^h-1\\ V_{0,h} & \text{if } 1\le a\le 2^h-2\wedge a\equiv 0 \bmod 4 \textbf{ or}\\ & \quad 2^h+2\le a\le 2^{h+1}-1\wedge a\equiv 3 \bmod 4\\ V_{5,h} & \text{if } a=2^h \end{cases}$$

Case 2: t is even but not 2

Step 1: select $c(x) = 1$

Select "ciphertext" $c(x) = 1$ and submit it to the decryption oracle. We obtain the polynomial $D(c(x) = 1) = v_0^{(1)} + v_1^{(1)}x + v_2^{(1)}x^2 + \cdots + v_{n-1}^{(1)}x^{n-1}$. Suppose there exists $v_i^{(1)} = t/2$. This means that either $u_i^{(1)} = \frac{t}{2}$ or $u_i^{(1)} = -\frac{t}{2}$. We want to find out which one among the two above cases holds.

1. If we are in case $u_i^{(1)} = \frac{t}{2}$, then we have $\left\lfloor \frac{t}{q} f_i \right\rceil = \frac{t}{2} \Leftrightarrow \frac{q}{2} - \frac{q}{2t} < f_i < \frac{q}{2}$
2. If we are in case $u_i^{(1)} = -\frac{t}{2}$, then we have $\left\lfloor \frac{t}{q} f_i \right\rceil = -\frac{t}{2} \Leftrightarrow -\frac{q}{2} < f_i < -\frac{q}{2} + \frac{q}{2t}$

To find out which one is the case, we have to wait for the next step.

Now, let's focus on all the other $v_i^{(1)} \neq \frac{t}{2}$. We have in this case, $v_i^{(1)} = u_i^{(1)}$. Now, similarly as before, we have $-\frac{t}{2} + 1 \leq u_i^{(1)} \leq \frac{t}{2}$, and every $u_i^{(1)}$ can have only t different values; it can be written as $u_i^{(1)} = -\frac{t}{2} + 1 + k_{i,1}, \quad$ with $k_{i,1} \in \{0, 1, \ldots, t-1\}$. Now, it is easy to see that

$$u_i^{(1)} = -\frac{t}{2} + 1 + k_{i,1} \Leftrightarrow -\frac{q}{2} + \frac{q}{t}(k_{i,1} + \frac{1}{2}) < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1} + \frac{3}{2})$$

The polynomial obtained from the decryption oracle can therefore be written as $D(c(x) = 1) = \sum_{i=0}^{n-1} \left(-\frac{t}{2} + 1 + k_{i,1} \right) x^i$. Each f_i belongs to the interval $(-q/2, q/2)$. But after this our first query we learn values $k_{i,1} \in [0, 1, \ldots, t-1]$, $0 \leq i \leq n-1$, such that $-\frac{q}{2} + \frac{q}{t}(k_{i,1} + \frac{1}{2}) < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1} + \frac{3}{2})$ We have that $-\frac{q}{2} + \frac{q}{t}(k_{i+1} + 3/2) - \left(-\frac{q}{2} + \frac{q}{t}(k_{i+1} + 1/2) \right) = \frac{q}{t}$. Therefore, we know each integer coefficient f_i with an error up to $\frac{q}{t}$.

The idea now is to keep submitting 'ciphertext' to the decryption oracle and obtain values $k_{i,j}$, with $0 \leq i \leq n-1$ and increasing integers $j = 1, 2, 3, \ldots$, in such a way that we keep reducing the interval in which f_i lies until we know f_i with an error smaller than 1, which determines each f_i completely.

Step 2: select $c(x) = 2$

Select now "ciphertext" $c(x) = 2 = 2 + 0x + \cdots + 0x^{n-1}$. Decryption oracle computes and return the polynomial

$$\begin{aligned} D(c(x) = 2) &= \left[\left\lfloor \frac{t}{q} \cdot [f \cdot 2]_q \right\rceil\right]_t \\ &= \left[\left\lfloor \frac{t}{q} \cdot \Big([2f_0]_q + [2f_1]_q x + \cdots + [2f_{n-1}]_q x^{n-1}\Big) \right\rceil\right]_t \end{aligned}$$

Now, let's focus on $\left[\left\lfloor \frac{t}{q}[2f_i]_q \right\rceil\right]_t x^i$ for each i such that, in the previous step, $v_i^{(1)} = \frac{t}{2}$.

1. We have

$$\begin{aligned} \frac{q}{2} - \frac{q}{2t} < f_i < \frac{q}{2} \;\; &\Leftrightarrow \;\; q - \frac{q}{t} < 2f_i < q \Leftrightarrow -\frac{q}{t} < [2f_i]_q < 0 \\ &\Leftrightarrow \;\; -1 < \frac{t}{q}[2f_i]_q < 0 \\ &\Leftrightarrow \;\; -1 \leq \left[\left\lfloor \frac{t}{q}[2f_i]_q \right\rceil\right]_t \leq 0 \\ &\Leftrightarrow \;\; \left[\left\lfloor \frac{t}{q}[2f_i]_q \right\rceil\right]_t = \begin{cases} 0 \text{ or } -1 & \text{if } t > 2 \\ 0 \text{ or } 1 & \text{if } t = 2 \end{cases} \end{aligned}$$

2. We have analogously $-\frac{q}{2} < f_i < -\frac{q}{2} + \frac{q}{2t} \Leftrightarrow \left[\left\lfloor \frac{t}{q}[2f_i]_q \right\rceil\right]_t = 0$ or 1.

From now on we assume $t > 2$; we will consider later the case in which $t = 2$. Let $v_i^{(2)} = \left[\left\lfloor \frac{t}{q}[2f_i]_q \right\rceil\right]_t$. We have that

1. if $v_i^{(2)} = -1$, then $u_i^{(1)} = \frac{t}{2}$ and $\frac{q}{2} - \frac{q}{2t} < f_i < \frac{q}{2}$
2. if $v_i^{(2)} = 1$, then $u_i^{(1)} = -\frac{t}{2}$ and $-\frac{q}{2} < f_i < -\frac{q}{2} + \frac{q}{2t}$
3. if $v_i^{(2)} = 0$, then we can't conclude right now the exact interval in which f_i belongs; this will be considered in the next step.

Remark 6. Suppose we are in the above case 3, i.e. $v^{(2)} = \left\lfloor \frac{t}{q}[2f_i]_q \right\rceil = 0$. Then

1. We have

$$\frac{q}{2} - \frac{q}{2t} < f_i < \frac{q}{2} \quad \wedge \quad \left\lfloor \frac{t}{q}[2f_i]_q \right\rceil = 0 \Leftrightarrow \frac{q}{2} - \frac{q}{4t} < f_i < \frac{q}{2}$$

2. Similarly, we have

$$-\frac{q}{2} < f_i < -\frac{q}{2} + \frac{q}{2t} \quad \wedge \quad \left\lfloor \frac{t}{q}[2f_i]_q \right\rceil = 0 \Leftrightarrow -\frac{q}{2} < f_i < -\frac{q}{2} + \frac{q}{4t}$$

We will use this remark in the next step to investigate further the interval in which f_i lies. Now, let's focus on all of the other coefficients. Using the same arguments as in section 5.10.2, the decryption oracle computes and return the polynomial

$$\begin{aligned}
D(c(x) = 2) &= \left[\left\lfloor \frac{t}{q} \cdot [f \cdot 2]_q \right\rceil\right]_t \\
&= \left[\left\lfloor \frac{t}{q} f_0^{(2)} \right\rceil + \left\lfloor \frac{t}{q} f_1^{(2)} \right\rceil x + \cdots + \left\lfloor \frac{t}{q} f_{n-1}^{(2)} \right\rceil x^{n-1} \right]_t \\
&= \left[u_0^{(2)} + u_1^{(2)} x + u_2^{(2)} x^2 + \cdots + u_{n-1}^{(2)} x^{n-1}\right]_t \\
&:= v_0^{(2)} + v_1^{(2)} x + \cdots + v_{n-1}^{(2)} x^{n-1}
\end{aligned}$$

As before, suppose there exists $v_i^{(2)} = t/2$. This means that either $u_i^{(2)} = \frac{t}{2}$, or $u_i^{(2)} = -\frac{t}{2}$. We can easily understand which case we are by considering the known value $v_i^{(1)} \neq \frac{t}{2}$. All the other $v_i^{(2)}$ correspond to values $u_i^{(2)} \neq \frac{-t}{2}$. These $u_i^{(2)}$ can then have only t different possible values, and can be written as $u_i^{(2)} = -\frac{t}{2} + 1 + k_{i,2}, \quad \text{with } k_{i,2} \in \{0, 1, \ldots, t-1\}$, and also

$$u_i^{(2)} = -\frac{t}{2} + 1 + k_{i,2} \Leftrightarrow -\frac{q}{2} + \frac{q}{t}(k_{i,2} + \frac{1}{2}) < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,2} + \frac{3}{2})$$

So now, for each $0 \leq i \leq n-1$ such that $v_i^{(1)} \neq \frac{t}{2} \vee (v_i^{(1)} = \frac{t}{2} \wedge v_i^{(2)} = 0)$, we know $k_{i,1}, k_{i,2}$ such that

$$\begin{cases} -\frac{q}{2} + \frac{q}{t}(k_{i,1} + \frac{1}{2}) < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,1} + \frac{3}{2}) \\ -\frac{q}{2} + \frac{q}{t}(k_{i,2} + \frac{1}{2}) < f_i < -\frac{q}{2} + \frac{q}{t}(k_{i,2} + \frac{3}{2}) \end{cases}$$

There are 3 cases to distinguish. These cases can be computed in an analogous way to what seen for the case t odd. We omit the details.

Generalization

We continue in this way, following the blueprint for t odd and taking care of all the coefficients for which $v_i^{(1)} = \frac{t}{2}$ and all subsequents $v_i^{(j)} = 0$ (when we finally find a $j \geq 2$ such that $v_i^{(j)} = 1$ or -1, then we can deduce the original value of $u_i^{(1)} = \frac{t}{2}$ or $-\frac{t}{2}$). If at the last step m we still get $v_i^{(m)} = 0$, then all the values $u_i^{(1)}$ remain undetermined, which also say that all the corresponding coefficients f_i can have only two possible values. At this point, the strategy is to submit to the decryption oracle 'ciphertexts' in order to determine whether $f_i \cdot f_j < 0$ or $f_i \cdot f_j > 0$ holds among all the non-zero coefficients f_i, f_j, in a way similar to what we have already discussed for the attack on the [55] SHW scheme. We omit the details; we will give a description of how to do this in the case $t = 2$; the general case $t > 2$ is then easy to obtain. We study now the case $t = 2$.

Case 3: t = 2

Step 1: select $c(x) = 1$

Choose and submit to the decryption oracle the polynomial $c(x) = 1$. It will compute and return the polynomial

$$D(c(x)=1) = \left[\left\lfloor \frac{2}{q} \cdot [f \cdot 1]_q \right\rceil\right]_2 = \left[\left\lfloor \frac{2}{q} f_0 \right\rceil + \left\lfloor \frac{2}{q} f_1 \right\rceil x + \cdots + \left\lfloor \frac{2}{q} f_{n-1} \right\rceil x^{n-1}\right]_2$$

For every $0 \leq i \leq n-1$, $u_i^{(1)} := \left\lfloor \frac{2}{q} f_i \right\rceil$ is such that $-1 \leq u_i^{(1)} \leq 1$, and so $v_i^{(1)} := [u_i^{(1)}]_2 = 0$ or 1. We have two cases to distinguish:

1) $v_i^{(1)} = 0$. We have $v_i^{(1)} = 0 \Leftrightarrow u_i^{(1)} = 0 \Leftrightarrow \left\lfloor \frac{2}{q} f_i \right\rceil = 0 \Leftrightarrow -\frac{1}{2} < \frac{2}{q} f_i < \frac{1}{2} \Leftrightarrow -\frac{q}{4} < f_i < \frac{q}{4}$

2) $v_i^{(1)} = 1$. We have

$$\begin{aligned} v_i^{(1)} = 1 \quad &\Leftrightarrow \quad u_i^{(1)} = -1 \text{ or } u_i^{(1)} = +1 \\ &\Leftrightarrow \quad \left\lfloor \frac{2}{q} f_i \right\rceil = -1 \text{ or } \left\lfloor \frac{2}{q} f_i \right\rceil = +1 \\ &\Leftrightarrow \quad -\frac{3}{2} < \frac{2}{q} f_i < -\frac{1}{2} \text{ or } \frac{1}{2} < \frac{2}{q} f_i < \frac{3}{2} \\ &\Leftrightarrow \quad -\frac{q}{2} < f_i < -\frac{q}{4} \text{ or } \frac{q}{4} < f_i < \frac{q}{2} \end{aligned}$$

Step 2: select $c(x) = 2$

Choose and submit to the decryption oracle the polynomial $c(x) = 2$. It will compute and return the polynomial

$$D(c(x)=2) = \sum_{i=0}^{n-1} \left[\left\lfloor \frac{2}{q} [2f_i]_q \right\rceil\right]_2 x^i =: \sum_{i=0}^{n-1} \left[u_i^{(2)}\right]_2 x^i =: \sum_{i=0}^{n-1} v_i^{(2)} x^i$$

We have two cases to distinguish:

1) $v_i^{(2)} = 0$. We have

$$\begin{aligned} v_i^{(2)} = 0 \;\; &\Leftrightarrow \;\; u_i^{(2)} = 0 \Leftrightarrow \left\lfloor \frac{2}{q}[2f_i]_q \right\rceil = 0 \Leftrightarrow -\frac{1}{2} < \frac{2}{q}[2f_i]_q < \frac{1}{2} \\ &\Leftrightarrow \;\; -\frac{q}{4} < [2f_i]_q < \frac{q}{4} \\ &\Leftrightarrow \;\; -\frac{q}{4} < 2f_i < \frac{q}{4} \text{ or } -\frac{5q}{4} < 2f_i < -\frac{3q}{4} \text{ or } \frac{3q}{4} < 2f_i < \frac{5q}{4} \\ &\Leftrightarrow \;\; -\frac{q}{8} < f_i < \frac{q}{8} \text{ or } -\frac{q}{2} < f_i < -\frac{3q}{8} \text{ or } \frac{3q}{8} < f_i < \frac{q}{2} \end{aligned}$$

We have three cases to distinguish, according to which known interval f_i lies at the end of step 1:

1.1) If $-\frac{q}{4} < f_i < \frac{q}{4}$, then $-\frac{q}{8} < f_i < \frac{q}{8}$

1.2) If $-\frac{q}{2} < f_i < -\frac{q}{4}$, then $-\frac{q}{2} < f_i < -\frac{3q}{8}$

1.3)] If $\frac{q}{4} < f_i < \frac{q}{2}$, then $\frac{3q}{8} < f_i < \frac{q}{2}$

2) $v_i^{(2)} = 1$. We have

$$\begin{aligned} v_i^{(2)} = 1 \;\; &\Leftrightarrow \;\; u_i^{(2)} = -1 \text{ or } u_i^{(2)} = +1 \\ &\Leftrightarrow \;\; \left\lfloor \frac{2}{q}[2f_i]_q \right\rceil = -1 \text{ or } \left\lfloor \frac{2}{q}[2f_i]_q \right\rceil = +1 \\ &\Leftrightarrow \;\; -\frac{3}{2} < \frac{2}{q}[2f_i]_q < -\frac{1}{2} \text{ or } \frac{1}{2} < \frac{2}{q}[2f_i]_q < \frac{3}{2} \\ &\Leftrightarrow \;\; -\frac{3q}{4} < [2f_i]_q < -\frac{q}{4} \text{ or } \frac{q}{4} < [2f_i]_q < \frac{3q}{4} \\ &\Leftrightarrow \;\; -\frac{3q}{4} < 2f_i < -\frac{q}{4} \text{ or } \frac{q}{4} < 2f_i < \frac{3q}{4} \\ &\Leftrightarrow \;\; -\frac{3q}{8} < f_i < -\frac{q}{8} \text{ or } \frac{q}{8} < f_i < \frac{3q}{8} \end{aligned}$$

Now, again we have three cases to distinguish, according to which known interval f_i lies at the end of step 1:

2.1) If $-\frac{q}{4} < f_i < \frac{q}{4}$, then $-\frac{q}{4} < f_i < -\frac{q}{8}$ or $\frac{q}{8} < f_i < \frac{q}{4}$

2.2) If $-\frac{q}{2} < f_i < -\frac{q}{4}$, then $-\frac{3q}{8} < f_i < -\frac{q}{4}$

2.3)] If $\frac{q}{4} < f_i < \frac{q}{2}$, then $\frac{q}{4} < f_i < \frac{3q}{8}$

Generalization and the last step

We continue in this way, and in the end we will know each coefficient f_i up to the sign. Therefore, we will know a polynomial $f'(x) = f'_0 + f'_1 x + \cdots + f'_{n-1}x^{n-1}$, with $f'_i = |f_i|$ for every i. We proceed similarly to what we have seen for the attack on the [55] scheme, i.e. we query the decryption oracle in order to find out the relations $f_i \cdot f_j < 0$ or $f_i \cdot f_j > 0$ among the coefficients f_i of the secret key $f(x)$. Suppose that the two consecutive coefficients f_i, f_{i-1} are both non-zero. We know their absolute values f'_i, f'_{i-1}. Choose and submit to the decryption oracle the polynomial $c(x) = \alpha|f_{i-1}| + \alpha|f_i|x$, with $\alpha \in (-q/2, q/2]$ such that $[2\alpha|f_{i-1} \cdot f_i|]_q \in \left[\frac{q}{4}, \frac{q}{2}\right]$ (it is always possible to find such an α). Now, the decryption oracle will compute and return the polynomial

$$D(c(x)) = \left[\left\lfloor \frac{2}{q}[\alpha|f_{i-1}|f_0 - \alpha|f_i|f_{n-1}]_q \right\rceil\right]_2 + \sum_{j=1}^{n-1} \left[\left\lfloor \frac{2}{q}[\alpha|f_{i-1}|f_j + \alpha|f_i|f_{j-1}]_q \right\rceil\right]_2 x^j$$

Let's focus on the i-th coefficient $\left[\left\lfloor \frac{2}{q}[\alpha|f_{i-1}|f_i + \alpha|f_i|f_{i-1}]_q \right\rceil\right]_2$. We have two cases:

1) If f_i, f_{i-1} have different signs, then

$$\alpha|f_{i-1}|f_i + \alpha|f_i|f_{i-1} = 0$$

and therefore the i-th coefficient is zero:

$$\left[\left\lfloor \frac{2}{q}[\alpha|f_{i-1}|f_i + \alpha|f_i|f_{i-1}]_q \right\rceil\right]_2 = 0$$

2) If f_i, f_{i-1} have the same positive sign, then

$$[\alpha|f_{i-1}|f_i + \alpha|f_i|f_{i-1}]_q = [2\alpha|f_i f_{i-1}|]_q \in \left[\frac{q}{4}, \frac{q}{2}\right]$$

In case f_i, f_{i-1} are both negative, we have that

$$[\alpha|f_{i-1}|f_i + \alpha|f_i|f_{i-1}]_q = [-2\alpha|f_i f_{i-1}|]_q \in \left[-\frac{q}{2}, -\frac{q}{4}\right])$$

In both cases, it easy to see that

$$\left[\left\lfloor \frac{2}{q}[\alpha|f_{i-1}|f_i + \alpha|f_i|f_{i-1}]_q \right\rceil\right]_2 = 1$$

So we can distinguish whether two consecutive non-zero coefficients f_i, f_{i-1} have the same sign or not. Exactly as we saw for the attack on the [55] scheme, this leads us to two possible candidates for the secret key; to determine which one is the correct one, it is enough to submit an extra appropriate query to the decryption oracle.

Remark 7. As we saw for the attack on the [55] scheme, we have to be careful in case one of the coefficient f_i is zero. In this case in fact, no information can be given about the sign of f_{i-1} if we compare it to f_i. To solve this problem, we have to choose and submit to the decryption oracle a polynomial in the form $c(x) = a + bx^j$, for appropriates a, b, j. We omit the details, which are straightforward from what we have just discussed and from the attack on the [55] scheme.

5.10.3 A Remark

Parallel to our work, Dahab, Galbraith and Morais [26] have also proposed similar attacks for [9, 55] but only for specific parameter settings at ICITS conference 2015. In comparison, our attacks apply to all parameter settings and are more efficient than theirs.

The key recovery attacks by Dahab, Galbraith and Morais [26] work for arbitrarily-tailored parameters for the LTV12 and BLLN13 SHE schemes. For example, they require $6(t^2+t) < q$ and $B^2 < \frac{q}{36t^2}$ while these conditions are not assumed in [55, 9]. On the other side, we hereby present attacks that work for all parameter settings. Moreover, our attacks are more efficient than theirs, see the following table. Note that n is defined as an integer of power of 2, B is a bound on the coefficient size of error distribution and is much smaller than q, $t \geq 2$ is an integer that partially determines the message space size. More detailed definitions for these parameters can be found in the following sections.

	Our Attacks	Attacks from [26]
[55]	$\lfloor \log_2 B \rfloor + n$	$n \cdot \lceil \log_2 B \rceil + n$
[9] (t is odd)	$\lceil \log_2 (B/t) \rceil$	$n \cdot \lceil \log_2 B \rceil$
[9] (t is even but not 2)	$\lceil \log_2 (B/t) \rceil + n$	$n \cdot \lceil \log_2 B \rceil$
[9] ($t = 2$)	$\lceil \log_2 (B/t) \rceil + n$	$n \cdot \lceil \log_2 B \rceil + n$

Chapter 6

Attacks to the modified LTV12 SHE scheme

We are going to analyze a line of attacks towards a modified version of the [55] SHE scheme. We refer to Section 5.9 for the details of the scheme.

As we already mentioned in Section 5.9, in [55], the authors do not explicitly state how the decryption behaves if μ mod 2 is not a constant. We can consider three scenarios: (1) output directly μ mod 2; (2) output the constant of μ mod R_2; (3) output an error. We have shown in Section 5.9 a key-recovery attack for scenario (1). We will see in this chapter how to extend this attack to scenario (2). It is likely that we can adapt our attack to scenario (3), but we have not succeeded so far. However, we there are good indicators that scenario (3) can resist to key-recovery attacks, and we are going to discuss this scenario in what follows.

6.1 Key-Recovery Attack: Scenario (2)

As we remarked earlier, the [55] SHE scheme only encrypts 1-bit plaintexts $m \in \{0, 1\}$, but decryption actually reveals a polynomial $\mu'(x) = \mu(x)$ mod

$2 = \mu'_0 + \mu'_1 x + \cdots + \mu'_{n-1} x^{n-1} \in R_2$. Therefore, if the ciphertext is honestly generated, we would expect $\mu'(x) \in \{0, 1\} \subset R_2$. Hence, a natural idea of preventing our attack would be to only output the coefficient μ'_0, since anyway $\mu'_i = 0$, for $1 \leq i \leq n-1$, if ciphertext is honestly generated. SHE scheme is the same as before, except the following alternative decryption step:

$\mathsf{Decrypt}'(\mathsf{sk}_1, \ldots, \mathsf{sk}_M, c)$**:**

- parse $\mathsf{sk}_i = f_i$ for $i \in [M]$
- let $\mu = f_1 \cdots f_M \cdot c \in R_q$
- $\mu'(x) := \mu(x) \bmod 2 = \mu'_0 + \mu'_1 x + \cdots + \mu'_{n-1} x^{n-1} \in R_2$
- output $\mu'_0 \in \{0, 1\}$

We show that, even in this case, our attack works with few modifications. Instead of recovering all coefficients s_i of the polynomial $s(x) = s_0 + s_1 x + \cdots + s_{n-1} x^{n-1} \in R_q$ at once, we are going to recover in sequence $s_0, s_1, \ldots, s_{n-1}$.

Recovering s_0

It is clear that, by performing the same attack as described above, we recover coefficient s_0 with at most N oracle queries; but no information will be leaked about s_i, for $1 \leq i \leq n-1$.

Recovering s_1

In order to recover s_1, we repeat the same attack as before, with the following modifications:

Preliminary step: submit to the decryption oracle the 'ciphertext' $c = -x^{n-1} \in R_q$. This way,

$$\begin{aligned} \mu(x) \bmod 2 &= s \cdot (-x^{n-1}) \bmod 2 = \\ &= (s_1 \bmod 2) + (s_2 \bmod 2)x + \cdots + \\ &\qquad \cdots + (s_{n-1} \bmod 2)x^{n-2} - (s_0 \bmod 2)x^{n-1} \end{aligned}$$

since $x^n = -1$ in R_q. So $\mu'_0 = s_1 \bmod 2$.

Similarly,

Step k, for $1 \leq k \leq m$, with $m = \lfloor \log_2(q-1) \rfloor$: Submit to the decryption oracle the 'ciphertext'

$$c = 2^k \cdot (-x^{n-1}) \in R_q$$

These modifications lead to a full recovery of s_1 (the final step is the same as in the original key-recovery attack).

Recovering s_i, for $0 \leq i \leq n-1$

Similarly, and more generally, we are going to recover $s_i \in [0, q-1]$, for all $0 \leq i \leq n-1$. Steps are as follows:

Preliminary step: submit to the decryption oracle the 'ciphertext' $c = -x^{n-i} \in R_q$.

Step k, for $1 \leq k \leq m$, with $m = \lfloor \log_2(q-1) \rfloor$: Submit to the decryption oracle the 'ciphertext'

$$c = 2^k \cdot (-x^{n-i}) \in R_q$$

Final step: same as in the original key-recovery attack.

6.2 Key-Recovery Attack: Scenario (3)

We are going here to investigate the modified [55] SHE scheme with ciphertext-validity check; is this possibly IND-CCA1-secure? Let's consider the following modified version of the SHE scheme in [55], where we tweak the decryption step. We focus here on a way of preventing our key-recovery attack, and possibly to make the [55] SHE scheme secure under key-recovery attacks. Decryption step is modified in order to perform a ciphertext-validity check before revealing the bit μ_0. Secret key sk is $f = 2f' + 1$ as in the original scheme.

$\mathsf{Decrypt}''(\mathsf{sk} = f, c)$**:**

- let $\mu(x) := (f \cdot c \in R_q) \bmod 2 = \mu_0 + \mu_1 x + \cdots + \mu_{n-1}x^{n-1} \in R_2$
- if $\mu(x) \notin \{0, 1\} \subset R_2$, return $\perp$
- else, output $\mu'(x) = \mu'_0 \in \{0, 1\}$

This simple ciphertext-validity check makes the scheme resistant to above key-recovery attack, and we could not yet find a working key-recovery attack against this modified SHE scheme. We tried to attack the scheme in such a way to gather the maximum possible information of the bits of the secret keys through repeted oracle decryption queries. Let's see the details.

Let $f' = f'_0 + f'_1 x + f'_2 x^2 + \cdots + f'_{n-1}x^{n-1} \in R$. We have

$$-\left\lfloor \frac{q}{2} \right\rfloor \leq |f'_i| \leq \left\lfloor \frac{q}{2} \right\rfloor$$

Since f' is B-bounded, we have in particular that

$$|f'_i| \leq B \ll q$$

And we have

$$f = 2f' + 1 = (2f'_0 + 1) + 2f'_1 x + 2f'_2 x^2 + \cdots + 2f'_{n-1} x^{n-1}$$

Since $B \ll q$, we still have $2B \ll q$.

Choose 'ciphertext' $c = 1$. We have

$$\begin{aligned} \mathrm{Dec}(2c) &= 2c * f = 2f \in R_q \\ &= (4f'_0 + 2) + 4f'_1 x + 4f'_2 x^2 + \cdots + 4f'_{n-1} x^{n-1} \\ &=: g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1} \end{aligned}$$

Since $|f_i| \leq B$ and $B \ll q$, we have $|f_i| \leq \frac{q}{2}$. So it is easy to see that $\mathrm{Dec}(2c) = 2f \bmod 2 = 0$.

Let's gradually increase the value $c \in \mathbb{N}$. Let $c_0 \in \mathbb{N}$ be the smallest integer greater than 1 such that

$$\mathrm{Dec}(2c_0) := g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1} \neq 0$$

Then, this means that $\exists i \in [0, 1, \ldots, n-1]$ such that $|g_i| > \left\lfloor \frac{q}{2} \right\rfloor$. Also, we have $\mathrm{Dec}(2(c_0 - 1)) = 0$, where

$$\mathrm{Dec}(2(c_0-1)) = [(4c_0-4)f'_0+(2c_0-2)]+(4c_0-4)f'_1 x+\cdots+(4c_0-4)f'_{n-1} x^{n-1}$$

And

$$\mathrm{Dec}(2c_0) = [4c_0 f'_0 + 2c_0] + 4c_0 f'_1 x + 4c_0 f'_2 x^2 + \cdots + 4c_0 f'_{n-1} x^{n-1}$$

So we have

1) $\mathrm{Dec}(2(c_0-1)) = 0$

2) $\mathrm{Dec}(2c_0) \neq 0$:

   2.1) $\mathrm{Dec}(2c_0) = 1$

   2.2) $\mathrm{Dec}(2c_0) = \perp$

From this we have that

$$1) \Rightarrow \begin{cases} \left|f'_0 + \frac{1}{2}\right| \leq \left\lfloor \frac{q-1}{8(c_0-1)} \right\rfloor \\ |f'_i| \leq \left\lfloor \frac{q-1}{8(c_0-1)} \right\rfloor \qquad \forall 1 \leq i \leq n-1 \end{cases}$$

$$2.1) \Rightarrow \begin{cases} \left|f'_0 + \frac{1}{2}\right| \geq \left\lceil \frac{q+1}{8c_0} \right\rceil \\ |f'_i| \leq \left\lfloor \frac{q-1}{8c_0} \right\rfloor \qquad \forall 1 \leq i \leq n-1 \end{cases}$$

$$2.2) \Rightarrow \begin{cases} \exists 1 \leq i \leq n-1 \text{s.t. } |f'_i| \geq \left\lceil \frac{q+1}{8c_0} \right\rceil \\ \left|f'_0 + \frac{1}{2}\right| \leq \left\lfloor \frac{q-1}{8c_0} \right\rfloor \text{ or } \left|f'_0 + \frac{1}{2}\right| \geq \left\lfloor \frac{q+1}{8c_0} \right\rfloor \end{cases}$$

(Notice that there may be more than one such coefficient f'_i.) So, to sum up:

First Step

- submit to the decryption oracle increasing integer values $c = 1, 2, 3, \ldots$
- let $c_0 \in \mathbb{N}$ be the smallest integer s.t. $\mathrm{Dec}(2c_0) \neq 0$. We have:

$$\mathrm{Dec}(2(c_0-1)) = 0 \Rightarrow \begin{cases} \left|f'_0 + \frac{1}{2}\right| \leq \left\lfloor \frac{q-1}{8(c_0-1)} \right\rfloor \\ |f'_i| \leq \left\lfloor \frac{q-1}{8(c_0-1)} \right\rfloor \qquad \forall 1 \leq i \leq n-1 \end{cases}$$

$$\mathrm{Dec}(2c_0) = 1 \Rightarrow \begin{cases} \left|f'_0 + \frac{1}{2}\right| \geq \left\lceil \frac{q+1}{8c_0} \right\rceil \\ |f'_i| \leq \left\lfloor \frac{q-1}{8c_0} \right\rfloor \qquad \forall 1 \leq i \leq n-1 \end{cases}$$

$$\mathrm{Dec}(2c_0) = \perp \Rightarrow \begin{cases} \exists 1 \leq i \leq n-1 \text{s.t. } |f'_i| \geq \left\lceil \frac{q+1}{8c_0} \right\rceil \\ \left|f'_0 + \frac{1}{2}\right| \leq \left\lfloor \frac{q-1}{8c_0} \right\rfloor \text{ or } \left|f'_0 + \frac{1}{2}\right| \geq \left\lfloor \frac{q+1}{8c_0} \right\rfloor \end{cases}$$

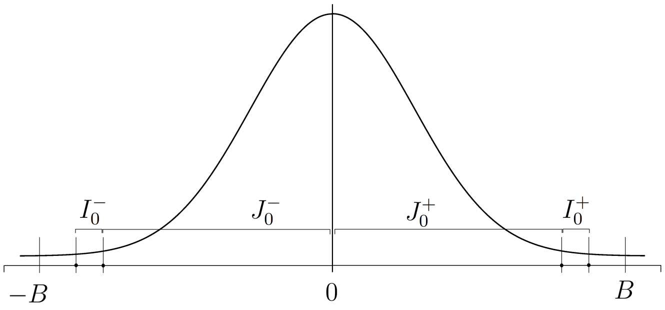
(Again, there may be more than such a coefficient f'_i)

Second Step

Define the intervals

$$\begin{aligned} I_0^+ &:= \left[\left\lceil \frac{q+1}{8c_0} \right\rceil, \left\lfloor \frac{q-1}{8c_0-8} \right\rfloor\right] \\ I_0^- &:= \left[-\left\lfloor \frac{q-1}{8c_0-8} \right\rfloor, -\left\lceil \frac{q+1}{8c_0} \right\rceil\right] \\ J_0^+ &:= \left[0, \left\lfloor \frac{q+1}{8c_0} \right\rfloor\right] \\ J_0^- &:= \left[-\left\lfloor \frac{q+1}{8c_0} \right\rfloor, 0\right] \end{aligned}$$

Notice that $\left\lfloor \frac{q-1}{8c_0-8} \right\rfloor \leq B$. We have



- Case 2.1) $\Leftrightarrow |f_0' + \frac{1}{2}| \in I_0^+$ and $|f_i'| \in J_0^+$, $\forall 1 \leq i \leq n-1$;
- Case 2.2) $\Leftrightarrow \exists i$s.t. $|f_i'| \in I_0^+$

Therefore, in Case 2.1) there is exactly one coefficient that overflows, but in Case 2.2) there may be more than one. Our next question is to compute the probability that such a case happens.

Question: In Case 2.2), what is the probability p_0 that $\exists! i \in [1, n-1]$ s.t. $|f_i'| \in I_0^+$, and also $|f_0' + \frac{1}{2}| \in J_0^+$?

Remember that the coefficients are chosen from the Gaussian normal distribution χ given by the function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}$$

where σ is the variance of the Gaussian distribution. We have, for a given $1 \leq j \leq n-1$, $j \neq i$:

$$P_j := \mathrm{Prob}(|f_j'| \in J_0^+, \exists j \neq i) = \frac{2}{\sigma\sqrt{2\pi}} \int_0^{\left\lfloor \frac{q-1}{8c_0-8} \right\rfloor} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x$$

Then, we have that the probability that every $1 \leq j \leq n-1$, $j \neq i$, belongs to J_0^+ is given by

$$P_{tot} := \mathrm{Prob}(|f_j'| \in J_0^+, \forall j \neq i) = P_j^{n-2}$$

Finally, we have

$$P_0 := \mathrm{Prob}(|f_0' + \frac{1}{2}| \in J_0^+) \approx P_j, \text{for some } j$$

In total,

$$p_0 = P_{\text{tot}} * P_0 \approx P_j^{n-1} = \left(\frac{2}{\sigma\sqrt{2\pi}} \int_0^{\left\lfloor \frac{q+1}{8c_0} \right\rfloor} \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \, \mathrm{d}x \right)^{n-1} \tag{6.1}$$

The probability $p_0 \approx P_j^{n-1}$ depends on the parameters σ, q, c_0.

a) If it is the Case 2.2) and at least two coefficients overflow, then it is difficult to continue the attack.

b) If it the Case 2.1) we have isolated the (only) coefficient which overflows (it is $|f_0'|$)

c) If it is the Case 2.2) and we have only one coefficient $|f_i'|$ overflowing, then we still don't know which one is it. However, it is easy to isolate which coefficient is it by simply query the decryption oracle with 'ciphertexts' $2c_0x^j$, for $j = 0, 1, 2, \ldots$. This simply shifts the position of the coefficient, until the coefficient $|f_i'|$ will be brought to position 0. More in particular, if the overflowing coefficient is $|f_i'|$, then $\text{Dec}(2c_0x^j) = \perp$ for $j = 0, 1, 2, \ldots, n-1-i$ and $\text{Dec}(2c_0x^j) = 1$ for $j = n-1-i+1$. So if j is the first value for which $\text{Dec}(2c_0x^j) = 1$, we know that $i = n - j$ and we have isolated the overflowing coefficient.

Therefore, if we are in cases b) or c) we can continue by submitting to the decryption oracle increasing values of $c_1 = c_0 + 1, c_0 + 2, \ldots$. Unfortunately, if we are in case a) we have no idea how to proceed: the ciphertext validity check in our tweaked decryption seems to make the scheme invulnerable to this line of key-recvoery attacks, which was successful for all the other SHE schemes.

Now, the probability that case 2.1) happens is given by p_0. In order to compute such probability, we need to compute formula 6.1, which depends on parameters σ, q, c_0 and B. If this probability is close to 1, then the proposed line of attack can be considered to a viable key recovery attack (we succeed in fact to isolate only one overflowing parameter with high probability and we can continue our attack). However, in order to actually compute this probability, we need concrete parameters to plug in in formula 6.1. Unfortunately, literature does not provide such concrete parameters, therefore we can only try to estimate them.

In [55], the authors state that ”Using the same security proof as in [74], we can base the security of the scheme [LTV12] on the DSPR assumption and the RLWE assumption. With the choice of parameters stated below, this yields security under the DSPR assumption and the hardness of approximating shortest vectors on ideal lattices to within a factor of 2^{n^ϵ}, which is believed to be hard.”

In the paper [55], the authors require that:

- $n = n(\lambda)$, where λ is the security parameter
- n is a power of 2
- $q = 2^{n^\epsilon}$, for $\epsilon \in (0, 1)$
- $r = \text{poly}(n)$
- $B = \text{poly}(n)$

Parameter Selection in the NTRU-FHE

For these reasons, it would be interesting to understand what concrete parameters to use in the [55] SHE scheme. Unfortunately There currently do not exist definitive, up to date works on these parameters.

We can consider then some parameters suggested for NTRU-FHE. Stehle and Steinfeld have shown that the DSPR problem is hard even for unbounded adversaries with their parameter selection. However, the new NTRU-FHE scheme will require different parameters to support homomorphic evaluation. The impact of the new parameter settings to the security level is largely unknown and requires careful research. However, even if we assume that the DSPR problem is hard for typical NTRU-FHE parameter selection, concrete parameters are still hard to chose. The RLWE problem is still relatively new and lacks thorough security analysis. A common approach is to assume that RLWE follows the same behavior as the LWE problem [43]. Under this assumption only, we can select parameters. If we omit the noise, given the prime number q and λ-bit security level, the dimension is bounded as in [43] as $n \leq \log(q)(\lambda+110)/7.2$. For example, given a 256-bit prime q, an 80-bit security level will require dimension $n = 6756$. ([31]).

To sum up, we can try to use the following parameters:

- $\lambda = 80$
- $n = 6756$
- $q = 2^{n^\epsilon}$ should have 256 bits. We can therefore take $q = 2^{256}$. Now since $\epsilon \in (0, 1)$ and q has 256 bits, we can take ϵ such that $n^\epsilon \approx 256$, i.e. $\epsilon \approx \log_{6756}(256) \approx 0.629$.
- c_0 is such that $B \leq \frac{q-1}{8c_0-8}$ and $B \geq \frac{q+1}{8c_0}$. From this, we get that $\frac{q}{8B} \leq c_0 \leq \frac{q-8B}{8B}$. We can assume that c_0 is in the middle of this interval, hence we can take $c_0 = \frac{q-4B}{8B}$.
- $B = \text{poly}(n) = n = 6756$ (totally arbitrary)

- $r = \mathrm{poly}(n) = n = 6756$ (totally arbitrary)

Using these parameters, we obtain

$$p_0 = \left(\frac{2}{\sigma\sqrt{2\pi}} \int_0^{\left\lfloor \frac{q+1}{8c_0} \right\rfloor} \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \,\mathrm{d}x \right)^{n-1} \approx 7.412860139583738 * 10^{-1121}$$

which basically is 0 probability. However, if we use as parameter B the following one

- $B = \mathrm{poly}(n) = n^2 = 6756^2 = 45,643,536$

we obtain

$$p_0 = \left(\frac{2}{\sigma\sqrt{2\pi}} \int_0^{\left\lfloor \frac{q+1}{8c_0} \right\rfloor} \mathrm{e}^{-\frac{x^2}{2\sigma^2}} \,\mathrm{d}x \right)^{n-1} \approx 1$$

Hence, until we don't obtain more concrete real parameters, nothing can be said about the the real potential of our line of attacks, since by tweaking parameters as we can in the given range, we can obtain both a working and a not working key recovery attack. This leaves us material to consider for an interesting future work.

Chapter 7

Conclusion and Future Work

Fully (and somewhat) homomorphic encryption is an interesting topic with potentially many useful applications. Current FHE schemes are far from being of any practical use, even though - as we have seen in Chapter 2 - there are already several real-world applications which make use of SHE schemes, and any implementer should keep in mind the threats of malicious attackers who can mount key-recovery attacks to these SHE schemes.

It is therefore important to have a clear and precise idea of what level of security is offered by the existing SHE and FHE schemes. This was the main goal of this dissertation: in Chapter 5 we have developed complete and efficient key-recovery attacks for most of the existing SHE schemes. More in particular, we showed that the SHE schemes from [15, 14, 12, 11, 44, 55, 9] suffer from key recovery attacks when the attacker is given access to the decryption oracle. Combining the results from [54, 76], we now know that most[1] existing SHE schemes suffer from key recovery attacks, and so they

[1] The only exception is the variation of [71] SHE scheme as shown in [54]. The authors use the notion of *valid ciphertexts*, and obtain an IND-CCA1 scheme. But as we already mentioned, this is obtained under a strong non-standard lattice knowledge assumption, and moreover they also show their scheme is not secure under a natural adaptive attack based on a *ciphertext validity oracle*. It would be desirable to have a IND-CCA1 secure SHE scheme which does rely on standard assumptions only.

are not IND-CCA1 secure. In Chapter 6 we have considered the SHE scheme from [55] - for which we developed an efficient key recovery attack in Chapter 5 - and we have tweaked its decryption step in two ways, leading to scenarios (2) and (3) as explained in that chapter. We successfully showed a key-recovery attack for scenario (2); however, scenario (3) seems to resist any attempt to show a key-recovery attack. We have shown that our usual strategy for key-recovery attacks does not lead to a successful attack, and our variation of the [55] SHE scheme seems therefore to be the a good candidate for being IND-CCA1 secure. As an interesting future work, one can look for a security proof of this scheme.

Our work sheds more light on the security of SHE schemes, and more importantly can be of great help to potential implementers who want to use a given SHE scheme for a real-world application. At the same time, our results encourage us and other researchers to investigate more on the IND-CCA1 security of SHE and FHE schemes, with the ultimate goal of obtaining a SHE scheme which offers this level of security, and relying on standard assumptions.

We now raise attention on some interesting future work and directions.

7.1 Future Directions

During our research on FHE, we crossed several interesting and important points which are worth of future investigation. We list here a few interesting future works, some of which are a natural direct continuation of our work.

Making IND-CCA1 secure SHE schemes based on standard assumptions

A major open problem is protect levelled homomorphic encryption from adaptive attacks that allow an adversary to learn the private key. Therefore, a natural next step is to investigate whether it is possible to enhance these SHE schemes to avoid key recovery attacks and make them IND-CCA1 secure. One thing we should keep in mind is to preserve their homomorphic properties. Following the work of [54], one could think of tweaking the decryption step of a SHE scheme by including a ciphertext validity check in order to make sure that, with some high probability, the ciphertext is honestly generated by the attacker and not specifically chosen for the purpose of recovering a given bit (or bits) of the secret key. Unfortunately, we cannot directly apply the techniques from [54] due to the fact that the SHE scheme from [54] enjoys some particular algebraic properties which do not exist in other schemes. So, we need to treat each SHE scheme individually.

At this moment it is still not clear whether we can adapt our key-recovery attack to the scenario (3) of the SHE scheme [55], but as noted in Chapter 6 we have good indicators that this can be. An interesting future work related to the above mentioned tweaked version of the [55] SHE scheme is to find a proof of its IND-CCA1 security.

Key recovery attacks and existing work on partial recovered secret key

Obtaining a SHE scheme which is IND-CCA1 secure has proved to be a difficult task, and so far no such scheme has been found. All the key-recovery attacks that we have developed, can completely determine each bit of the secret key sk. For the way they have been developed, these attacks work by querying the decryption oracle in such a way that for each oracle reply, a new bit of the secret key is determined. The secret key is then fully recovered once a given amount of oracle queries is performed. (In general, these key recovery attacks require a number of oracle queries approximatively equal to the number of bits of the secret key.)

An interesting future direction would therefore be to focus our attention to a more practical way of preventing key recovery attacks. Consider the scenario in which we consider a malicious attacker allowed to submit only a limited number of queries to the decryption oracle, and therefore he's unable to recover fully the secret key of the SHE solely relying on oracle decryption queries. (See also Remark 3.) We could thus address the question on how many bits of the secret key can be leaked, while preserving the IND-CCA1 security of a given SHE scheme; i.e. to what extent a given SHE scheme is secure under this attack, which we call *partial key-recovery attack*. How can we extract the secret key in case we can recover only a limited number of bits of the secret key through direct queries to the decryption oracle?

More precisely, let $n \in \mathbb{N}$ be the number of the bits of sk. Let $r \leq n$ be the number of bits of sk determined by the attacker after each oracle query, and let t be the number of oracle queries needed by the attacker to recover all the bits of sk. In most of the attacks that we previously, we have $r = 1$ and $t \approx n$, so that after approximatively n queries the attacker can fully determine sk. In our new scenario, we have to solve the following problem:

Problem. Consider a given SHE scheme $\mathscr{E}$ whose secret key sk has n bits; in binary notation, we write $\mathsf{sk} := s = (s_{n-1}s_{n-2}\cdots s_1 s_0)_2$. Consider an attacker $\mathscr{A}$ who wants to break its IND-CCA1 security by recovering sk. Let's assume that with each oracle query Q_i, for $i = 1, 2, \ldots$, $\mathscr{A}$ is learning a single different bit s_i of sk. Let $t' \leq n$ be the maximum number of queries that $\mathscr{A}$ can submit to the decryption oracle.
Determine what is the minimum value $0 \leq t' \leq n$ allowing $\mathscr{A}$ to recover sk without extra queries.

Other directions

Among other interesting open problems, we can consider the following one recently proposed by S. Galbraith et al. in [53]. The authors investigated a different countermeasure to our key recovering attack by considering how key recovery attacks could be prevented in a scenario in which a SHE scheme has more than one valid secret key, and for every decryption a different secret key is used. More in particular, the idea is to generate a ”one-time” private key every time the decryption algorithm is run, so that even if an attacker can learn some bits of the one-time private key from each decryption query, this does not allow them to compute a valid private key. They show an implementation of their idea on the [44] SHE scheme. This new approach is trying to achieve security against adaptive attacks which does not rely on a notion of ”valid ciphertexts” (see also Remark 2).

This way, key recovery attacks (in the style we developed them in this thesis, i.e. recovering the secret key fully bit-by-bit) seem to fail in some particular cases.

References

[1] Ajtai, M. (1996). Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA. ACM.

[2] Ajtai, M. and Dwork, C. (1997). A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 284–293, New York, NY, USA. ACM.

[3] Ajtai, M., Kumar, R., and Sivakumar, D. (2001). A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 601–610, New York, NY, USA. ACM.

[4] Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C. A., and Strand, M. (2015). A guide to fully homomorphic encryption. Cryptology ePrint Archive, Report 2015/1192. http://eprint.iacr.org/2015/1192.

[5] Bellare, M. and Palacio, A. (2004). Towards plaintext-aware public-key encryption without random oracles. In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62.

[6] Bellare, M. and Rogaway, P. (1995). *Optimal asymmetric encryption*, pages 92–111. Springer Berlin Heidelberg, Berlin, Heidelberg.

[7] Bernstein, D. J., Chuengsatiansup, C., Lange, T., and van Vredendaal, C. (2016). Ntru prime. Cryptology ePrint Archive, Report 2016/461. http://eprint.iacr.org/2016/461.

[8] Boneh, D., Halevi, S., Hamburg, M., and Ostrovsky, R. (2008). Circular-secure encryption from decision diffie-hellman. In Wagner, D., editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer Berlin Heidelberg.

[9] Bos, J. W., Lauter, K., Loftus, J., and Naehrig, M. (2013). Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, LNCS. Springer.

[10] Bos, J. W., Lauter, K., and Naehrig, M. (2014). Private predictive analysis on encrypted medical data. Cryptology ePrint Archive, Report 2014/336. http://eprint.iacr.org/2014/336.

[11] Brakerski, Z. (2012). Fully homomorphic encryption without modulus switching from classical gapsvp. In Safavi-Naini, R. and Canetti, R., editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886.

[12] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2012). (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325. ACM.

[13] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., and Stehlé, D. (2013). Classical hardness of learning with errors. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 575–584, New York, NY, USA. ACM.

[14] Brakerski, Z. and Vaikuntanathan, V. (2011a). Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011*, pages 505–524.

[15] Brakerski, Z. and Vaikuntanathan, V. (2011b). efficient fully homomorphic encryption from (standard) lwe. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 97–106.

[16] Brakerski, Z. and Vaikuntanathan, V. (2014). Lattice-based fhe as secure as pke. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, New York, NY, USA. ACM.

[17] Chen, Y. and Nguyen, P. Q. (2012). Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'12, pages 502–519.

[18] Chenal, M. and Tang, Q. (2014). On key recovery attacks against existing somewhat homomorphic encryption schemes. In *LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 239–258. Springer.

[19] Chenal, M. and Tang, Q. (2015). Key recovery attack against an ntru-type somewhat homomorphic encryption scheme. In Lopez, J. and Mitchell, J. C., editors, *Information Security: 18th International Conference, ISC 2015*, pages 397–418. Springer International Publishing.

[20] Cheon, J., Coron, J.-S., Kim, J., Lee, M., Lepoint, T., Tibouchi, M., and Yun, A. (2013). Batch fully homomorphic encryption over the integers. In Johansson, T. and Nguyen, P. Q., editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 315–335.

[21] Chung, K.-M., Kalai, Y., and Vadhan, S. (2010). Improved delegation of computation using fully homomorphic encryption. Cryptology ePrint Archive, Report 2010/241. http://eprint.iacr.org/2010/241.

[22] Cohn, H. and Heninger, N. (2011). Approximate common divisors via lattices. Cryptology ePrint Archive, Report 2011/437. http://eprint.iacr.org/2011/437.

[23] Coron, J.-S., Mandal, A., Naccache, D., and Tibouchi, M. (2011). Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology - CRYPTO 2011*, pages 487–504.

[24] Coron, J.-S., Naccache, D., and Tibouchi, M. (2012). Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2012*, pages 446–464.

[25] Cramer, R. and Shoup, V. (1998). *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, pages 13–25. Springer Berlin Heidelberg, Berlin, Heidelberg.

[26] Dahab, R., Galbraith, S., and Morais, E. (2015). Adaptive key recovery attacks on ntru-based somewhat homomorphic encryption schemes. In *Information Theoretic Security - 8th International Conference, ICITS*, volume 9063 of *LNCS*, pages 283–296. Springer.

[27] Damgard, I. (1991). Towards practical public-key schemes secure against chosen ciphertext attacks.

[28] Damgård, I., Pastro, V., Smart, N., and Zakarias, S. (2012). *Multiparty Computation from Somewhat Homomorphic Encryption*, pages 643–662. Springer Berlin Heidelberg, Berlin, Heidelberg.

[29] Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654.

[30] Ding, J. and Tao, C. (2014). A new algorithm for solving the approximate common divisor problem and cryptanalysis of the fhe based on gacd. IACR Cryptology ePrint Archive, Report 2014/042.

[31] Doroz, Y., Hu, Y., and Sunar, B. (2014). Homomorphic aes evaluation using ntru. Cryptology ePrint Archive, Report 2014/039. http://eprint.iacr.org/.

[32] ElGamal, T. (1985). *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, pages 10–18. Springer Berlin Heidelberg, Berlin, Heidelberg.

[33] Galbraith, S. D., Gebregiyorgis, S. W., and Murphy, S. (2016). Algorithms for the approximate common divisor problem. Cryptology ePrint Archive, Report 2016/215. http://eprint.iacr.org/2016/215.

[34] Gama, N. and Nguyen, P. Q. (2008). Predicting lattice reduction. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pages 31–51, Berlin, Heidelberg. Springer-Verlag.

[35] Gennaro, R. and Wichs, D. (2012). Fully homomorphic message authenticators. Cryptology ePrint Archive, Report 2012/290. http://eprint.iacr.org/2012/290.

[36] Gentry, C. (2009a). *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA.

[37] Gentry, C. (2009b). Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178. ACM.

[38] Gentry, C. (2010). Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105.

[39] Gentry, C. and Halevi, S. (2011a). Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 107–109.

[40] Gentry, C. and Halevi, S. (2011b). Implementing gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology - EUROCRYPT 2011*, pages 129–148.

[41] Gentry, C., Halevi, S., and Smart, N. P. (2012a). Better bootstrapping in fully homomorphic encryption. In *Proceedings of the 15th International Conference on Practice and Theory in Public Key Cryptography*, PKC'12, pages 1–16.

[42] Gentry, C., Halevi, S., and Smart, N. P. (2012b). Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology - EUROCRYPT 2012*, pages 465–482.

[43] Gentry, C., Halevi, S., and Smart, N. P. (2012c). Homomorphic evaluation of the aes circuit. Cryptology ePrint Archive, Report 2012/099. http://eprint.iacr.org/.

[44] Gentry, C., Sahai, A., and Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Canetti, R. and Garay, J., editors, *Advances in Cryptology - CRYPTO 2013*, volume 8042 of *LNCS*, pages 75–92.

[45] Goldreich, O., Goldwasser, S., and Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 112–131, London, UK, UK. Springer-Verlag.

[46] Gorbunov, S., Vaikuntanathan, V., and Wichs, D. (2015). Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 469–477, New York, NY, USA. ACM.

[47] Group, T. T. W. (February 2013). The notorious nine: Cloud computing top threats in 2013.

[48] Hoffstein, J., Pipher, J., and Silverman, J. (1998). Ntru: A ring-based public key cryptosystem. In Buhler, J., editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer Berlin Heidelberg.

[49] Howgrave-Graham, N. (2001). *Cryptography and Lattices: International Conference, CaLC 2001 Providence, RI, USA, March 29–30, 2001 Revised Papers*, chapter Approximate Integer Common Divisors, pages 51–66.

[50] Lauter, K. (2015). Practical applications of homomorphic encryption. In *EUROCRYPT 2015*.

[51] Lauter, K., López-Alt, A., and Naehrig, M. (2015). *Private Computation on Encrypted Genomic Data*.

[52] Lenstra, A., Lenstra, H.W., J., and Lovász, L. (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534.

[53] Li, Z., Galbraith, S. D., and Ma, C. (2016). *Preventing Adaptive Key Recovery Attacks on the GSW Levelled Homomorphic Encryption Scheme*, pages 373–383. Springer International Publishing.

[54] Loftus, J., May, A., Smart, N. P., and Vercauteren, F. (2012). On cca-secure somewhat homomorphic encryption. In *Proceedings of the 18th International Conference on Selected Areas in Cryptography*, SAC'11, pages 55–72.

[55] López-Alt, A., Tromer, E., and Vaikuntanathan, V. (2012). On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1219–1234. ACM.

[56] Lyubashevsky, V., Peikert, C., and Regev, O. (2010). On ideal lattices and learning with errors over rings. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 1–23, Berlin, Heidelberg. Springer-Verlag.

[57] Micciancio, D. (2001). Improving lattice based cryptosystems using the hermite normal form. In *Revised Papers from the International Conference on Cryptography and Lattices*, CaLC '01, pages 126–145, London, UK, UK. Springer-Verlag.

[58] Micciancio, D. and Goldwasser, S. (2002). *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston, Massachusetts.

[59] Micciancio, D. and Peikert, C. (2011). Trapdoors for lattices: Simpler, tighter, faster, smaller. IACR Cryptology ePrint Archive, Report 2011/501.

[60] Micciancio, D. and Voulgaris, P. (2010). A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 351–358, New York, NY, USA. ACM.

[61] Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124.

[62] Nathan Dowlin, Ran Gilad-Bachrach, K. L. K. L. M. N. J. W. (2015). Manual for using homomorphic encryption for bioinformatics.

[63] Nguyen, P. (1999). Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto '97. In Wiener, M., editor, *Advances in Cryptology — CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. Springer Berlin Heidelberg.

[64] Nguyen, P. Q. and Valle, B. (2009). *The LLL Algorithm: Survey and Applications*. Springer Publishing Company, Incorporated, 1st edition.

[65] Nuida, K. (2014). A simple framework for noise-free construction of fully homomorphic encryption from a special class of non-commutative groups. IACR Cryptology ePrint Archive, Report 2014/097.

[66] Paillier, P. (1999). *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg.

[67] Peikert, C. (2009). Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA. ACM.

[68] Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93.

[69] Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978a). On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179.

[70] Rivest, R. L., Shamir, A., and Adleman, L. (1978b). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

[71] Smart, N. P. and Vercauteren, F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, PKC'10, pages 420–443.

[72] Smart, N. P. and Vercauteren, F. (2011). Fully homomorphic simd operations.

[73] Stehle, D. and Steinfeld, R. (2010). Faster fully homomorphic encryption. In Abe, M., editor, *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 377–394.

[74] Stehlé, D. and Steinfeld, R. (2011). *Advances in Cryptology – EUROCRYPT 2011: 30th Annual International Conference on the Theory and*

Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, chapter Making NTRU as Secure as Worst-Case Problems over Ideal Lattices. Springer Berlin Heidelberg.

[75] van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT 2010*, pages 24–43.

[76] Zhang, Z., Plantard, T., and Susilo, W. (2012). On the cca-1 security of somewhat homomorphic encryption over the integers. In *Proceedings of the 8th International Conference on Information Security Practice and Experience*, ISPEC'12, pages 353–368.

Appendix A

Publications

- [18] Chenal M., Tang Q. (2014) On Key Recovery Attacks Against Existing Somewhat Homomorphic Encryption Schemes. In: Aranha D., Menezes A. (eds) Progress in Cryptology - LATINCRYPT 2014. LATINCRYPT 2014. Lecture Notes in Computer Science, vol 8895. Springer, Cham

- [19] Chenal M., Tang Q. (2015) Key Recovery Attacks Against NTRU-Based Somewhat Homomorphic Encryption Schemes. In: Lopez J., Mitchell C. (eds) Information Security. ISC 2015. Lecture Notes in Computer Science, vol 9290. Springer, Cham