University of Wollongong

LATTICE-BASED CRYPTOGRAPHY: A PRACTICAL IMPLEMENTATION

A Thesis Submitted in Partial Fulfilment of
the Requirements for the Award of the Degree of

Master of Computer Science - Research

from

UNIVERSITY OF WOLLONGONG

by

Michael Rose
BCompSci (Digital Systems Security)

School of Computer Science and Software Engineering
Faculty of Informatics

2011

CERTIFICATION

I, Michael Rose, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Master of Computer Science - Research, in the School of Computer Science and Software Engineering, Faculty of Informatics, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

(Signature Required)

Michael Rose
06 Oct 2011

Table of Contents

Lattice-based Cryptography: A practical implementation

Michael Rose

A Thesis for Master of Computer Science - Research

School of Computer Science and Software Engineering
University of Wollongong

ABSTRACT

Ever since Ajtai's seminal paper *Generating Hard Instances of Lattice Problems* [4] there has been much interest in developing Lattice-based cryptography for a number of reasons. Firstly, Lattice-based cryptography provides a much stronger notion of security, in that the average-case of certain problems in lattice-based cryptography are equivalent to the worst-case of those problems [4]. Secondly, there are strong indications that these problems will remain secure under the assumption of the availability of quantum computers, unlike both the integer-factorisation and discrete-logarithm problems as relied upon in many conventional cryptosystems. [72]

Despite these advantages however, a significant hurdle to the widespread adoption of lattice-based cryptography has been that most lattice-based cryptosystems are computationally expensive and hence impractical compared to conventional cryptosystems. In this dissertation, the author will explore various methods to improve the practicality of lattice-based cryptosystems and to optimise the algorithms that make up these cryptosystems for modern computer processors.

KEYWORDS: Lattice, Cryptography, Security, Encryption, Decryption, Signature, CRT, GGH, Micciancio

Acknowledgements

The author would like to thank the following people for their help and advice: Dr. Thomas Plantard, Prof. Willy Susilo, Dr. Victor Shoup, Dr. Damien Stehlé, Loukas Kalenderidis, Bill Tourloupis, Graeme Phillips, Tim Broady, Anastasia Mikuscheva, Natalie Farrawell, Elise Jackson.

The author would also like to apologise to all those who have had the dubious privelege of entertaining his animated discussions on the topic of lattice research with feigned interest.

Chapter 1

Introduction

Albert Einstein's assertion that Quantum Entanglement as a theory was incorrect as it resulted in 'spooky action at a distance' was indicative of the resistance of conventional physicists to the notion of quantum mechanics in the first half of the 20th century. It is precisely this property of quantum particles however, that brings great hope for new and much more efficient computing algorithms. The use of this quantum entanglement property for computing purposes dates back to 1982 [24] and since then, interest has been growing at a phenomenal rate, due in part to the effect quantum computers will have on conventional cryptography.

1.0.1 Quantum computing

A quantum computer consists of a number of *qubits* (quantum bits) together with some basic qubit gates and a mechanism for brief storage of these qubits. The primary difference that qubits have over standard conventional bits is that qubits can be expressed probabilistically, having any real probability between 0 and 1, unlike a conventional bit which must have only one of two states. Once the qubit is read however, the quantum state collapses into one of these two states. [72] This property of having an indeterminate state before being read, together with the entanglement of multiple qubits, allows a new class of algorithms to be developed enabling a quantum computer to perform some specific computations exponentially faster than a traditional computer. Due to this new paradigm of computation, a quantum computer cannot be modelled as a conventional Turing machine efficiently [21].

1.0.1.1 Quantum complexity classes

Since a quantum computer cannot be efficiently modelled as a conventional Turing machine, new complexity classes need to be introduced. The BQP (Bounded error Quantum Polynomial) complexity class is informally analogous to the conventional P (Polynomial) complexity class. ie. a class of problems that are considered feasible to compute on a quantum computer, albeit allowing some error due to a small, non-zero probability of quantum decoherence induced errors and the probabilistic nature of quantum calculations. Similarly, the QMA (Quantum, Merlin-Arthur) complexity class is roughly analogous to the conventional NP (Non-deterministic Polynomial) complexity class, being considered as the class of problems containing infeasible algorithms to compute on a quantum computer. It is suspected that these classes are not equal, however it is important to note that the relationships between these two complexity classes are not proven, just as it is not known whether P = NP.

1.0.1.2 Implications of quantum computing

In 1994, Shor developed algorithms that can factor integers and solve the Discrete Logarithm problem both in BQP [72]. This was groundbreaking for the field of conventional cryptanalysis as the ability to solve these two problems in polynomial time broke the long held complexity assumptions (and hence security) that almost all major asymmetric-key cryptosystems at the time relied upon. Further development into quantum algorithms resulted in Grover's algorithm in 1996 [34] which reduces the time of an unsorted database search by a quadratic factor over the fastest possible algorithm on a conventional computer. This algorithm (and related algorithms) can be used to invert *any* secure one-way function faster than a conventional computer, given enough space. While the existence of this algorithm does not inherently break conventional symmetric cryptosystems based on one-way functions, in certain cases such cryptosystems could be rendered insecure.

1.0.2 Quantum Key Distribution

In 1984, Bennett and Brassard created the first Quantum Key Distribution (QKD) protocol; BB84 [12]. This protocol is an unconditionally secure mechanism by which two parties can agree on a common key. Such a key can then be used to form a one-time pad, creating an information-theoretically secure cryptosystem. Unfortunately such an approach has significant drawbacks, primarily the requirement for a new quantum-aware infrastructure to support such quantum transmissions. Since all QKDs created thus far are point-to-point session key exchanges, the feasibility of such a Quantum Key Infrastructure would rely on enormous advances in Quantum Storage devices. Additionally, due to the extremely fragile quantum carriers used (whether they are photons, electrons, ions etc.), such an infrastructure would need to overcome two significant hurdles. Firstly, this infrastructure would have to be kept in an extremely controlled state, as any unintended interaction with the quantum information carrier would cause irreversible quantum decoherence, significantly reducing throughput and greatly increasing overhead in error correction. Secondly, to prevent man-in-the-middle attacks, the systems involved must communicate through an authenticated channel, using on average, singular quantum information carriers, uninterrupted from source to destination. This limits the feasibility of routing networks with large physical size requirements. Unfortunately, it also seems that in the literature, the creation of such an authenticated channel for use in a QKD network is an exercise left to the reader. If such an authentication mechanism is based on a classical algorithm (such as password-authenticated Diffie-Hellman or PKI signatures) then any proposed security benefit disappears when viewed in the context of quantum computer cryptanalysis.

Chapter 2

Post-Quantum Cryptography

Due to the limited scope and significant drawbacks involved with implementing efficient Quantum Key Distribution, the search for a conventional, non-quantum cryptography solution that will work in existing infrastructures is a rapidly growing area of research. Such research has been given the term Post-Quantum Cryptography, post-quantum indicating that these cryptographic constructs are believed to remain secure after practical, large scale quantum computers become available. Primarily, cryptosystems based on one-way trapdoors not believed to be in BQP but can operate efficiently on conventional computing platforms are sought. Four general areas of research have emerged:

- Hash-based signatures
- Code-based cryptography
- Multivariate cryptography
- Lattice-based cryptography

Each approach has unique advantages and disadvantages, which will be explored in the following sections.

2.1 Hash-based Signatures

Traditional signature schemes have relied on number-theoretic problems to ensure security; primarily relying on the hardness of integer factorization and discrete logarithm. Since both of these problems cannot be used in any secure post-quantum signature scheme, a significant body of research has been produced on the topic of hash-based signatures. An enormous benefit these signature schemes boast over traditional signatures is that they are not being tied to any specific hash function. Such modularity is rarely seen in an asymmetric cryptosystem. Should a hash function be broken or compromised in any way, it is a simple matter to change to a hash function that is still presumed secure.

2.1.1 Hard problems in Hashing

Hash functions are specifically designed to be resistant to certain operations. Different scenarios require different security assurances to be present in the hash function; a hash function designed for one scenario may not be suitable for use in another. The three primary security goals are described below.

Definition 1 (Pre-Image Resistance). *Given a pre-image resistant hash function* $H : \{0,1\}^* \rightarrow \{0,1\}^k$ *and a message hash* h*, it is computationally infeasible to generate a valid message* m *such that* $H(m) = h$*.*

Definition 2 (Second Pre-Image Resistance). *Given a second pre-image resistant hash function* $H : \{0,1\}^* \rightarrow \{0,1\}^k$ *and a message* m_1*, it is computationally infeasible generate a valid message* $m_2 \neq m_1$ *such that* $H(m_1) = H(m_2)$*.*

Definition 3 (Collision Resistance). *Given a collision-resistant hash function* $H : \{0,1\}^* \rightarrow \{0,1\}^k$*, it is computationally infeasible to generate a message pair* $m_1 \neq m_2$ *such that* $H(m_1) = H(m_2)$*.*

2.1.2 Hash-based one-time signatures

The Lamport-Diffie One-Time Signature scheme (LD-OTS) [42] was proposed by Lamport in 1979 as a signature scheme that is provably secure, requiring only that the hash function used is collision resistant and each public/private key-pair is only used to sign one message.

Key generation

1. A collision resistant hash function g is chosen, $g : \{0,1\}^* \rightarrow \{0,1\}^n$.
2. A one-way function f is chosen, $f : \{0,1\}^n \rightarrow \{0,1\}^n$.
3. Signing key X is created as two sets of n uniformly random bit-strings B, each bit-string being length n.
$$X = \{X_0, X_1\},$$
$$X_k = \{B_{k,1}, B_{k,2}, ..., B_{k,n}\}, \quad B_{k,i} \in \{0,1\}^n.$$
4. Verification key Y is created in a similar manner, with each bit-string being the corresponding bit-string of X passed through one-way function f.
$$Y = \{Y_0, Y_1\},$$
$$Y_k = \{C_{k,1}, C_{k,2}, ..., C_{k,n}\}, \quad C_{k,i} = f(B_{k,i}).$$

Signing

1. A digest d of message M is generated using the collision resistant hash-function $d = g(M)$.
2. Signature S is constructed by choosing n bit-strings from the signature key X

in a consecutive manner, from the subset X_0 or X_1 based on the corresponding bit of digest d.

$$S = \{S_0, S_1, ..., S_n\},$$

$$S_i = B_{d[i],i}, \quad 1 \leq i \leq n,$$

Where $d[i]$ denotes the i'th bit of digest d.

Verification

1. The verifier calculates the digest d of the message M, $d = g(M)$.
2. Each bitstring in S is passed through the one-way function f and compared to the corresponding bitstring from Y, with the set Y_0 or Y_1 chosen based on the corresponding bit in digest d.

$$\forall 1 \leq i \leq n, f(S_i) \overset{?}{=} C_{d[i],i}.$$

3. If and only if all bitstrings match, the signature is valid.

It can be seen that a single use of this signature reveals half the signing key. As such, both keys must be discarded after use as the signing key can no longer be trusted. While LD-OTS is quite efficient to compute, the signatures produced are relatively large. An improvement was proposed by Winternitz in 1979 [50] that allows each bitstring in X to sign several digest bits, reducing the signature size dramatically. For an in-depth analysis we refer the reader to [23].

2.1.3 The Merkle Signature Scheme

Since the creation and distribution of keys is needed every time a one-time signature is used, the practicality of these schemes are questionable. Merkle's proposed solution

[50] in 1979 attempts to alleviate this problem by creating a large number of these keys, bound together in a tree structure. Given any collision resistant cryptographic hash function $g : \{0,1\}^* \Rightarrow \{0,1\}^n$ and an existing secure one-time signature scheme, the Merkle Signature Scheme (MSS) operates as follows.

Key Generation

1. The signer selects a height value H of the tree, $H \in \mathbb{Z}$, $H \geq 2$.
2. The signer generates 2^H key-pairs for the chosen one-time signature scheme.
3. The signer creates a binary tree of height H and sets each leaf node to the hash of each separate verification key.
4. The internal nodes of this tree then are constructed from the leaf to the root, taking the value of the hash of the concatenation of the child nodes.
5. The root is given as the signer's MSS public key.

Signing

1. Given a message M, generate digest $d = g(M)$.
2. Create signature S of digest d with first available one-time signing key.
3. Create authenticity chain a of the corresponding one-time verification key v, by traversing the tree from the verification leaf to the root, recording the sibling of each node in the path.
4. Send $\{M, S, v, a\}$.

Verification

1. Verify that the authenticity chain a is valid by comparing each node to the hash of it's two children and the root to the signer's MSS public key.

2. Verify that S is a valid signature of M using the one-time signature verification key v.

2.1.4 Evaluation

Given a hash function that is provably collision resistant (and hence, also pre-image resistant) then Merkle's signature scheme with Lamport-Diffie one-time signatures is provably secure under an adaptive chosen message attack (CMA) [16]. The efficiency of the standard MSS scheme as described by Merkle in 1979 is quite low, considering the number of hash calculations needed in the key-generation and verification stages. In addition, the storage requirements for the one-time keys is enormous for any practical public-key scenario. Improvements have been made on Merkle's scheme, such as the use of space-time trade-offs in the tree traversal and the use of Pseudo-Random Number Generators to generate the signature keys [16]. Particularly promising is the adaptation of tree chaining to Merkle's scheme by Buchmann et. al [15], whereby one of the signature key pairs is used to sign a new tree root (and hence new public key). A signature key-pair of this tree is then used to sign the next tree as it begins to approach capacity. By continually signing new trees in this way, the security of the original scheme is retained while capacity is effectively infinite.

Since the security of both the one-time signature scheme and the Merkle Signature Scheme is reliant on the security of the underlying hash function, it is noted that for the scheme to be ultimately provably secure, the hash function must also be provably collision resistant. While provably secure hash functions do exist, many of these provably secure hash functions are based on assumptions that are not resistant against a quantum adversary. However, some hash functions have been designed to be provably secure in the quantum sense (such as SWIFFT [44] and AFS [10]). While using these hash functions with an efficient form of MSS may seem like the perfect post-quantum signature scheme; and indeed is provably secure under the lattice assumption; several code-based and lattice-based signature schemes will be described in the following sections which do not suffer the overhead that using a post-quantum hash with a hash-based signature scheme causes.

2.2 Code-based cryptography

Coding theory has a rich history in information theory as a mechanism to remove inherent data redundancy from a message (ie. source coding) or to ensure it's transmission error free (channel coding). With the explosion of radio communications in the 20th century, together with both World Wars, research into coding theory increased substantially. Not only was it important to reduce data redundancy as much as possible to compress messages, likewise it was extremely important for these messages to reach their recipients error free.

It is primarily the error correcting codes developed for use in channel coding that also find use in code-based cryptography, the first system being the McEliece cryptosystem developed in 1978, using the presumed difficulty of decoding randomized Goppa codes [48].

2.2.1 Binary Linear Codes

The simplest error correcting code is a repeating code. In this simple scheme, each bit is repeated, at least twice for single-bit error detection and at least three times for single-bit error correction. The repetition code is extremely inefficient however, and many better codes have been developed since. A class of codes, referred to as linear codes, operate as follows.

Code Set-up

1. A code $\mathcal{C}$ with block input length k and codeword length $n \geq k$ is defined by a k-row, n-column binary generator matrix $G \in \mathbb{F}_2^{k,n}$, where each row of G is a separate codeword of the code $\mathcal{C}$.
2. G is permuted into systematic form. ie. $G = (I_k|P)$.
3. A parity check matrix H is calculated $H = (-P^T|I_{n-k})$.
4. $d \in \mathbb{Z}$ is defined as the minimum hamming distance of the code $\mathcal{C}$ (ie. the minimum number of bits that are different between any two codewords in $\mathcal{C}$).
5. It can be seen that the maximum number of bits capable of being corrected is:
$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$
6. A set $\mathcal{S}$ is computed of all possible error permutations strictly less than or equal to t bits multiplied by the parity check matrix. ie.
$$\mathcal{S} = (e, He) : \quad e \in \mathbb{F}_2^n, \quad \sum_{i=0}^{n} e_i \leq t.$$

Encoding

1. A codeword x is generated by multiplying some input vector v with the generator matrix. ie. $x = vG$.

Decoding (Syndrome-decoding method)

1. A received codeword z is defined as the transmitted codeword x with some error vector e added. ie. $z = x + e$.

2. The syndrome vector s is defined as the parity check matrix H multiplied by the received codeword z. ie. $s = Hz$.

3. It can be seen that the parity check matrix multiplied with any codeword $x \in \mathcal{C}$ will result in the zero vector. As such, we have:

$$s = Hz = H(x + e) = Hx + He = 0 + He = He.$$

4. s is compared against the set $\mathcal{S}$ and the resulting e is derived.

5. The original codeword x is computed by removing the error vector e from the received codeword z. ie. $x = z - e$.

The primary difference between linear codes is the method in which the generator matrix is chosen. Binary Goppa codes for instance are constructed by using an algebraic genus-0 curve X over a finite field $\mathbb{F}_2$.

2.2.2 The McEliece Cryptosystem

McEliece published a paper [48] in 1978 describing a new public-key cryptosystem based on binary irreducible Goppa codes. Although any error-correcting code class can be used as the one-way trapdoor function, many other codes have been broken [60]. The original proposition of Goppa codes has not been broken to date [60]. The structure of the McEliece class of cryptosystems is as follows.

Key Generation

1. A generator matrix G for a code $\mathcal{C}$ is created, with a parameter t such that:

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

2. The public key $G^{PUB} = TGP$, where T is a random, binary, non-singular matrix and P is a random permutation matrix.

3. The private key is $\{T, D_{\mathcal{C}}, P\}$, where $D_{\mathcal{C}}$ is an efficient decoding algorithm for $\mathcal{C}$. eg. A set $\mathcal{S}$ for use with syndrome decoding.

Encryption

1. A plaintext message vector m is multiplied with G^{PUB}. ie. $x = mG^{PUB}$.

2. A small error vector e of hamming weight t is xored to make decryption non-trivial and this is sent as the ciphertext.

$$z = x + e.$$

Decryption

1. The ciphertext is multiplied with the inverse of the permutation matrix P. ie. $y = zP^{-1}$.

2. The error vector e is then removed by applying the algorithm $D_{\mathcal{C}}$ to y.

3. Finally, the plaintext is recovered by multiplying by the inverse of T.

2.2.3 Evaluation

Due to the large memory requirements for binary Goppa codes of an adequate security parameter [60], many other error-correcting codes have been used with McEliece with varying success. Many have been proven insecure and modifications have consequently been made to the McEliece cryptosystem to account for the new cryptanalysis attacks [59] [74]. Research is continuing into this area of post-quantum cryptography however, not just in producing an efficient, secure, public key cryptosystem, but also into code-based cryptographic hash functions [10] and pseudo-random number generators [25].

2.3 Multivariate cryptography

Owing to their impressive speed, multivariate-based cryptosystems are showing much promise as a class of practical post-quantum cryptosystems. In general, given the results of a set of multivariate equations, it is NP-hard to determine the structure of the multivariate equations [22]. However, due to the fact that a trapdoor must be embedded in the system for it to be viable as a public key cryptosystem, this removes the guarantee that the problem is NP-Hard. Many cryptosystems developed based on multivariate one-way trapdoor functions which were originally presumed secure have since been broken.

A set $\mathcal{P}$ of multivariate polynomials is defined,

$$\mathcal{P} = \{p_1(x_1, \dots, x_n), \dots, p_m(x_1, \dots, x_n)\}, \quad x_k \in \mathbb{F}_q,$$

where each polynomial takes the form:

$$p_k(x_1, \dots, x_n) = \sum_i P_{ik} x_i + \sum_i Q_{ik} x_i^2 + \sum_{i>j} R_{ijk} x_i x_j.$$

2.3.1 Multivariate cryptosystems

In general, if a set of multivariate polynomials is evaluated with a single input vector $X = (x_1, x_2, \cdots, x_n)$, with the output of each polynomial given in a vector $C = \{p_1(X), p_2(X), \cdots, p_m(X)\}$, it is difficult to find the original vector X, given only C. In the standard bipolar multivariate cryptosystem, the set of multivariate polynomials $\mathcal{P}$ is not taken at random however; it is created by taking a system $\mathcal{Q}$ that belongs to a class of multivariate polynomials known to be easy to invert, together with two affine maps. These maps serve to transform the system into one that is difficult to invert, thereby creating a trapdoor. This general idea of taking some information in which an operation is easy to invert and then perturbing the information in such a way that the operation is hard to invert closely parallels the McEliece cryptosystem. Indeed, the primary difference between various multivariate cryptosystems lies not in the basic structure, but in the choice of map $\mathcal{Q}$.

Key generation

1. A system of multivariate polynomials $\mathcal{Q}$ is created such that it is easy to invert.
2. Two affine maps S and T are created.
3. The map $\mathcal{P}$ is calculated by $\mathcal{P} = \mathcal{S} \circ \mathcal{Q} \circ \mathcal{T}$.
4. $\mathcal{P}$ is published as the public key.
5. $\{\mathcal{S}, \mathcal{Q}, \mathcal{T}\}$ is retained as the private key.

Encryption

1. A plaintext vector p is used as the input of each polynomial in the public multivariate system $\mathcal{P}$.

2. A ciphertext vector c consisting of each of the outputs of the polynomial functions is calculated.

Decryption

1. A vector u is calculated from applying the ciphertext vector c to the inverse of the affine map $\mathcal{T}$. ie. $u = T^{-1}(c)$.
2. A vector v is calculated from applying the vector u to the inverse of the central map $\mathcal{Q}$. ie. $v = Q^{-1}(u)$.
3. The plaintext vector p is recovered by applying the vector v to the inverse of the affine map $\mathcal{S}$. ie. $p = S^{-1}(v) = S^{-1}(Q^{-1}(T^{-1}(c)))$.

2.3.2 Construction Methods

Since the construction of the central map $\mathcal{Q}$ is central to the security and efficiency of the cryptosystem, many construction methods have been devised. It is important to note that most of these listed have been broken in the original parameter specification.

Small-field approaches

A Triangular map is defined as:

$$J(x_1, ..., x_n) = (x_1 + g_1(x_2, ..., x_n), ..., x_{n-1} + g_{n-1}(x_n), x_n)$$

where the g_i are arbitrary polynomial functions. If these functions are known, then J is invertible [22]. Unfortunately these maps are able to be attacked using *rank* attacks (see [20] and [33]).

Another small-field construction is that of Oil and Vinegar schemes [62]. This involves splitting the n variables into two sets, such that we have o *oil* variables and

$v = n - o$ *vinegar* variables. A map $\mathcal{Q} : \mathbb{F}_q^n \mapsto \mathbb{F}_q^o$ is then created, such that each polynomial of the central map is of the form:

$$1 \leq l \leq o, q_l(x) = \sum_{i=1}^{v} \sum_{j=1}^{n} \alpha_{i,j}^{(l)} x_i x_j,$$

where all coefficients α are randomly chosen from $\mathbb{F}_q$.

As there are no quadratic terms of oil variables, it can be seen that the oil variables and the vinegar variables don't fully mix (referring to a salad dressing, explaining the name of the scheme). This scheme is easily inverted by guessing values for the vinegar variables, which results in a set of o linear equations with the oil variables. This set has a high probability of having a solution, however if it has no solution, new vinegar variables are guessed and the resulting set of linear equations tested again for a solution. By using the affine transforms specified above, the variables are mixed such that an attacker is unsure as to which were the vinegar variables and which were the oil variables. Unfortunately this scheme has been attacked for when $o = v$. By choosing $v \neq o$, we have an Unbalanced Oil and Vinegar (UOV) scheme [40], which seems secure for around two to three times the number of vinegar variables than oil variables [22].

Big-field approaches

A *big-field* approach to multivariate cryptography is where the central map $\mathcal{Q}$ is embedded in a finite extension field $\mathbb{F}_{q^n}$, rather than the original finite field $\mathbb{F}_q$. This allows the central map to be inverted easily, as long as it is of a certain structure.

The Matsumoto-Imai construction [47] involves the creation of a central map of the following form:

$$\mathcal{Q} : \mathbf{x} \in \mathbb{F}_{q^n} \mapsto \mathbf{y} = \mathbf{x}^{1+q^\alpha},$$

where $\gcd(1 + q^{\alpha},\, q^n - 1) = 1$.

The Hidden Field Equation (HFE) class of derivatives is a natural extension of the Matsumoto-Imai scheme, whereby the central map is of the form:

$$\mathcal{Q} : \mathbf{x} \in \mathbb{F}_{q^n} \mapsto \mathbf{y} = \sum_{0 \le i,j < n} a_{i,j}\mathbf{x}^{q^i + q^j} + \sum_{0 \le i < n} b_i \mathbf{x}^{q^i} + c.$$

This central map can be inverted using the Berlekamp algorithm [13], with time complexity $O(nd^2 \log d + d^3)$ [22].

2.3.3 Evaluation

If the system is operated in a base field of $\mathbb{F}_2$ then we see extremely fast cryptosystem speeds and small keysizes compared to other post-quantum cryptosystems. Unfortunately many multivariate cryptosystems have been broken and to date, there appears to be no practical multivariate cryptosystem developed that inspires a high security confidence in the research community [22]. An ongoing research effort into finding a trapdoor that is resistant to both conventional and quantum attacks is needed for multivariate cryptosystems to be seen as a secure, practical method for post-quantum cryptography.

2.4 Lattice-based cryptography

Since Ajtai's seminal paper *"Generating hard instances of Lattice problems"* [5], Lattice-theory has emerged as a significant research area into efficient, provably secure post-quantum cryptography. The security assurances given by lattice-based cryptography provide a much greater confidence in the long-lasting security of cryptosystems built using hard lattice problems for two reasons. Firstly, many problems in lattice-theory are proven to be NP-Hard [14] (ie. at least as hard as the hardest problems in NP).

While it is still unknown (and indeed may never be known) the relation between the post-quantum classes of BQP and QMA; and conventional classes P and NP; we do know that these problems are at least as hard as any problem in NP, which is a much stronger assertion than can be made for most problems conventional cryptosystems are based on. Secondly, the security of many lattice problems have a worst-case to average-case reduction [5]. This reduces the requirement of a cryptosystem's security proof to a proof of average-case hardness due to this worst-case to average-case reduction. This equivalence also allows claims about the security of superclasses of lattice problems to be made, if we know the security of a lattice problem within the superclass. Informally, if a particular instance of a lattice problem is known to be hard, any superclass containing this instance is known to be at least as hard on average. This provides greater flexibility in designing cryptosystems to better suit the needs of the scenario.

A full-rank *lattice basis* $\mathbf{B}$ is defined as a set of n linearly independent vectors in a vector space of dimension n.

$$\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}, \quad \mathbf{b}_k \in \mathbb{R}^n.$$

A *lattice* $\mathcal{L}_\mathbf{B}$ is defined as the set of all the integral combinations of a basis $\mathbf{B}$ of linearly independent vectors across a vector space of dimension n.

$$\mathcal{L}_\mathbf{B} = \mathbb{Z}\,\mathbf{b}_1 + ... + \mathbb{Z}\,\mathbf{b}_n, \quad \mathbf{b}_k \in \mathbb{R}^n.$$

Some problems in lattice-based cryptography are easy to solve using bases of a particular structure. We refer to *good* bases as those in which a given problem is easy to solve, and *bad* bases as those in which it is generally no easier than a random basis to solve a particular lattice problem. Since each lattice may be instantiated by an infinite number of bases, the ability to create a lattice that can be instantiated by both a *good* basis and a *bad* basis simultaneously is the fundamental premise upon which many one-way trapdoors are built.

2.4.1 Lattice-based Cryptosystems

With Ajtai's seminal paper in 1996 [5], it was shown that a cryptosystem can be created with its security proof reliant only on the worst-case of the one-way trapdoor function through the *hidden hyperplane problem.* While this introduced the notion of the worst-case/average-case reduction, due to the enormous ciphertext expansion and the keysizes involved for an adequate security level, this cryptosystem was not ever intended to be a practical replacement of existing cryptosystems. Furthermore, Nguyen and Stern presented a heuristic attack against this cryptosystem [58]. Since then, this initial proposition has been improved [31, 17, 39, 28, 77, 76, 46] and inspired many other cryptosystems based on SVP [66, 67, 6].

The cryptosystem developed by Goldreich, Goldwasser and Halevi in 1996 [30] was a step closer to a practical lattice-based cryptosystem. Vastly improving on Ajtai's extreme ciphertext expansion, this cryptosystem sparked a sustained interest in developing a practical cryptosystem using integral lattices and further improvements were made (See Micciancio [51], Plantard et al. [63], Rose et al. [69]). The basic structure of these cryptosystems follow the design of the McEliece cryptosystem with a lattice trapdoor rather than a code trapdoor. The original GGH cryptosystem is described as follows:

Key Generation

1. Create a *good* basis $\mathbf{R}$.

2. Transform this *good* basis $\mathbf{R}$ into a *bad* basis $\mathbf{Q}$ through a unimodular transfor-

mation.

3. Publish *bad* basis $\mathbf{Q}$ as public basis and keep *good* basis $\mathbf{R}$ as private basis.

Encryption

1. Choose any lattice vector $\mathbf{w}$ using the public basis $\mathbf{Q}$ and add some small plaintext vector $\mathbf{p}$ to it.

2. Send this new vector $\mathbf{c} = \mathbf{w} + \mathbf{p}$ as the ciphertext.

Decryption

1. Using the private basis, compute the closest lattice vector $\mathbf{w}$ to the ciphertext $\mathbf{c}$.

2. Subtract this lattice vector $\mathbf{w}$ from the ciphertext to give the plaintext $\mathbf{p} = \mathbf{c} - \mathbf{w}$.

It is important to note that while Nguyen broke the original GGH cryptosystem in 1999 due to a limited parameter set, the basic premise is still viable.

2.4.2 Evaluation

With the worst-case/average-case reduction shown by Ajtai and Dwork [7], together with some lattice problems shown to be NP-Hard [14], lattice-based cryptography shows much promise for a practical, secure post-quantum cryptosystem. While many lattice-based cryptosystems boast simplicity and elegance, the computational complexity is still relatively high compared to both conventional cryptosystems as well as some multivariate cryptosystems. Indeed, it would almost appear as though lattice-based cryptographic research is a race towards efficiency whereas multivariate-based cryptographic research is a race towards security. With constructions such as q-ary lattices and the ideal lattice classes, this efficiency gap is closing quickly.

2.5 Summary

While hash-based signatures show promise for a viable post-quantum signature scheme due to their good security reduction, particularly when used with provably collision-resistant post-quantum hash functions (such as AFS [10], SWIFFT [45] and SWIFFTX [9]), the large key and signature sizes still currently render hash-based signatures impractical. This area of research is also unlikely to present a post-quantum alternative to conventional encryption schemes.

The code-based McEliece cryptosystem has not yet been broken, despite not having any formal security proof. Since the McEliece cryptosystem, at least using the paremeter set described by McEliece, is inefficient and has a large memory requirement, many derivatives have been constructed attempting to alleviate these drawbacks. Unfortunately many derivatives of the McEliece cryptosystem have been broken however and it seems unlikely that the security assumptions made by code-based cryptography will have a known relation with a formal complexity class since the main security assumption underlying code-based cryptography has no proof of hardness (unlike basic multivariate and lattice problems). This lack of proof makes it possible that an algorithm could be developed (conventional or quantum) that renders the fundamental problem behind code-based cryptosystems insecure.

While Multivariate-based cryptography is very fast and has small keys, many multivariate-based cryptosystems have been broken and the security confidence in new multivariate systems is lacking. While solving multivariate equation systems is NP-Hard in general, the adaptation of this problem to allow polynomial time decryption does not appear to be well-understood due to the many attacks on these schemes. A multivariate-based cryptosystem with a tight security proof based on a well-founded security assumption is needed before security confidence in multivariate-based cryptosystems will reach a point that would support practical adoption.

Lattice-based cryptography shows much promise where other post-quantum cryptographic systems lack, such as the variety of applications that lattice-based cryptographic systems can be developed for, the availability of many lattice problems (many of which are reducible to NP-Hard) and the astonishing worst-case/average-case reduction shown by Ajtai. More research is needed however, to bridge the gap between known NP-Hard problems and the variants that are used in lattice-based cryptography as well as improving the practicality of these schemes.

For a tabled summary of the four post-quantum cryptographic research areas, together with a subjective analysis, refer to Table 2.1.

	Hash-based	**Code-based**	**Multivariate-based**	**Lattice-based**
Schemes	Signature	Signature Encryption Hash	Signature Encryption	Signature Encryption Hash Oblivious Transfer Identity-Based Encryption Homomorphic Encryption
Security reduction	Collision Resistance	Code invertibility	Solving Multivariate equation system	Finding good basis for a lattice Solving lattice problems in special lattices
Theoretic Speeds	Dependent on hash function used	Good for Hardware	Good for Hardware	Good for Software
Practical Speeds	Extremely Fast	Good	Untested	Untested
Advantages	Extremely fast with good security reduction Extremely modular	Mature with first scheme remaining secure	Fast Small keysizes	Excellent security reductions
Disadvantages	Only signature Relies on secure hash function Large signatures	Many variants proven insecure Secure variants have extensive memory requirements	Low security confidence due to many systems broken	Not fully understood

Table 2.1: Comparison among different techniques for post-quantum cryptography

Chapter 3

Lattice Theory

3.1 Overview

In the following chapters, we will use a standard notation. *Small* scalar integers and rationals will be represented in the lowercase roman alphabet (such as k and n). *Large* scalar integers and rationals will be represented in the uppercase roman alphabet (such as M and P). Scalar real numbers will be represented in the lowercase greek alphabet (such as γ and ϕ). Vectors will be represented by boldface lowercase roman letters (such as $\mathbf{v}$ and $\mathbf{w}$. Matrices will be represented as boldface uppercase roman letters (such as $\mathbf{B}$ and $\mathbf{H}$).

A lattice is defined informally as the set of all integral linear combinations of a set of basis vectors in an n-dimensional vector space. It follows that from a geometrical perspective, this produces a set of regular, repeating points of a set pattern. Similarly, it also follows that such a lattice can be defined by an infinite number of bases. Multiple bases defining the same lattice are said to exhibit lattice equality. A formal definition can be seen in Def. 4.

Definition 4 (Lattice)**.** *A lattice $\mathcal{L}$ is a discrete sub-group of $\mathbb{R}^n$, or equivalently the set of all the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$.*

$$\mathcal{L}_{\mathbf{B}} = \mathbb{Z}\,\mathbf{b}_1 + \cdots + \mathbb{Z}\,\mathbf{b}_n, \quad \mathbf{b}_i \in \mathbb{R}^n.$$

$\mathbf{B} = (\mathbf{b}_1, ..., \mathbf{b}_n)$ *is called a basis of* $\mathcal{L}_{\mathbf{B}}$ *and* d*, the dimension of* $\mathcal{L}_{\mathbf{B}}$*. We will refer* $\mathcal{L}_{\mathbf{B}}$ *as a lattice of basis* $\mathbf{B}$.

3.2 Lattice properties

3.2.1 Rank

The *rank* of a lattice is defined as the number of linearly independent vectors in any basis for that lattice. A lattice that is *full-rank* is defined as a lattice where the number of linearly independent vectors in any basis for this lattice is exactly equal to the number of dimensions in which the lattice is embedded. In such an instance, it is clear that any basis for such a lattice can be described by a set of n vectors, each of n dimensions. We can thus describe the basis as a square integer matrix, in *row-vector* form. In this thesis, unless otherwise specified, we will only be operating with full-rank lattices, with bases described by square matrices in row-vector form.

3.2.2 Determinant

In this dissertation, we refer to *lattice* determinants (as opposed to matrix determinants) to measure properties about the lattice. Specifically, we define $det(\mathcal{L}_{\mathbf{B}})$ as being the n-dimensional volume of the fundamental parallelpiped defined by the lattice basis $\mathbf{B}$. Since we will only be operating with full-rank lattices in this thesis, we can simplify the definition of this lattice determinant as being the absolute value of the determinant of some basis of the lattice. ie.

$$det(\mathcal{L}_{\mathbf{B}}) = |det(\mathbf{B})|, \quad \mathbf{B} \in \mathbb{Z}^{n,n}$$

.

Any multiplication of a lattice basis with a unimodular matrix will produce a new basis that generates the same lattice. In fact, lattice equality is only achieved if there exists such a unimodular transform between bases. [2] Due to the determinant of a unimodular matrix being ± 1, it is clear that the choice of basis will not affect the lattice determinant. As such, the choice of basis for a particular lattice has no effect on the volume of the fundamental parallelotope and we refer to the lattice determinant as an *invariant* of the lattice.

3.2.3 Norms

Many problems in lattice theory involve distance minimization. While the most intuitive way to measure distance in a multi-dimensional space is by using the *Euclidean Norm*, other norms exist. Unless otherwise specified, in this thesis we will operate exclusively with the Euclidean norm.

Euclidean Norm

The *Euclidean norm* is the most intuitive norm to use in 2-dimensional and 3-dimensional spaces. This norm comes from Pythagoras' theorem, stating that the distance between two points is the square root of the sum of the axial distances squared. This can be extended to an arbitrary, finite-dimensioned vector space by squaring each of the axial dimensions and taking the square root of the sum.

Definition 5 (Euclidean norm)**.** *Let* $\mathbf{w}$ *be a vector of* $\mathbb{R}^n$*. The Euclidean norm is the function* $\|.\|_2$ *defined by*

$$\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{n} |\mathbf{w}_i|^2}$$

Taxi-cab Norm

The *Taxi-cab norm* or the *Manhattan norm* is named due to the rectangular grid of city streets (eg. Manhattan) that a taxi or indeed any wheeled road vehicle must legally follow to travel between two points. This norm is defined as the sum of the axial distances.

Definition 6 (Taxi-cab norm)**.** *Let* $\mathbf{w}$ *be a vector of* $\mathbb{R}^n$*. The* L_1 *norm is the function* $\|.\|_1$ *defined by*

$$\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |\mathbf{w}_i|$$

p-Norm

Both of these norms can be generalized into a parametrized version. This is referred to as the p-norm.

Definition 7 (p-norm)**.** *Let* $\mathbf{w}$ *be a vector of* $\mathbb{R}^n$*. The p-norm is the function* $\|.\|_p$ *defined by*

$$\|\mathbf{w}\|_p = \sqrt[p]{\sum_{i=1}^{n} |\mathbf{w}_i|^p}$$

Infinite Norm

The p-norm can be taken at the limit case as p approaches infinity, giving us the *max*-norm or the *infinite*-norm.

Definition 8 (infinite-norm)**.** *Let* $\mathbf{w}$ *be a vector of* $\mathbb{R}^n$*. The max norm is the function* $\|.\|_\infty$ *defined by*

$$\|\mathbf{w}\|_\infty = max(|\mathbf{w}_1|, |\mathbf{w}_2|, ..., |\mathbf{w}_n|)$$

3.2.4 Minima

The minimum λ_i of a lattice $\mathcal{L}$ is defined as the radius of the smallest *hypersphere*, centred at the origin, containing i linearly independent lattice vectors. Unless otherwise stated, the radius will be measured using the Euclidean norm.

3.2.5 Unimodular Matrices

If a basis $\mathbf{B}$ can be transformed by a multiplication with a transformation matrix $\mathbf{U}$ such that the new basis $\mathbf{B}'$ yields the same lattice as the original basis $\mathbf{B}$ (ie. $\mathbf{B}' = \mathbf{U} \times \mathbf{B} \quad \mathcal{L}_{\mathbf{B}} \equiv \mathcal{L}_{\mathbf{B}'}$), we refer to this transformation $\mathbf{U}$ as a *unimodular* transformation. [2]

A unimodular transformation matrix is defined as an integer matrix, whose inverse is also integral. This implies the following properties:

1. $\mathbf{U}$ must be integral.

2. $\mathbf{U}$ must be square.

3. $det(\mathbf{U})$ must be exactly ± 1.

As such, any basis of a lattice can be transformed to any other basis for the same lattice through a multiplication with a single unimodular transformation matrix.

3.2.6 Hermite Normal Form

If a matrix $\mathbf{H}$ is of Hermite Normal Form, then the matrix $\mathbf{H}$ must adhere to the following criteria:

- $\mathbf{H}$ contains no negative coefficients.

- $\mathbf{H}$ is triangular.

- $\mathbf{H}$ diagonals are strictly greater than all other elements in the column.

A more formal definition is described in Def. 9.

Definition 9 (HNF). *Let $\mathcal{L}$ be an integer lattice of dimension d and $\mathbf{H} \in \mathbb{Z}^{d,n}$ a basis of $\mathcal{L}$. $\mathbf{H}$ is a Hermite Normal Form basis of $\mathcal{L}$ if and only if:*

$$\forall 1 \leq i, j \leq d \quad \mathbf{H}_{i,j} \begin{cases} = 0 & \textit{if } i > j \\ \geq 0 & \textit{if } i \leq j \\ < \mathbf{H}_{j,j} & \textit{if } i < j \end{cases}$$

For any matrix $\mathbf{B}$, there exists some unimodular matrix $\mathbf{U}$ such that $\mathbf{H} = \mathbf{U} \times \mathbf{B}$ where $\mathbf{H}$ is of Hermite Normal Form [51]. As such, there exists a function $f(\mathbf{B})$ that decomposes some matrix $\mathbf{B}$ into a Hermite Normal Form,

$$\mathbf{H} = f(\mathbf{B}), \quad \mathcal{L}_{\mathbf{H}} \equiv \mathcal{L}_B$$

.

Furthermore, the HNF of a lattice basis is unique for the lattice [51]; any basis representing the same lattice will have the same HNF decomposition. Indeed, unlike the lattice determinant, the HNF of any lattice basis completely describes that lattice. Polynomial time algorithms exist for the decomposition of a matrix into HNF [51].

3.2.7 Reducing a vector modulo a lattice

In many lattice problems, we consider the relation between a vector $\mathbf{v} \in \mathbb{Z}^n$ and some lattice $\mathcal{L}_{\mathbf{B}}$. Computationally, since we represent the lattice $\mathcal{L}_{\mathbf{B}}$ as the basis matrix $\mathbf{B}$, we are actually only interested with the relationship between the vector $\mathbf{v}$ and the basis $\mathbf{B}$ translated to the vicinity of $\mathbf{v}$. Since the lattice's domain is infinite, we can simplify this relation by translating the vector $\mathbf{v}$ to the vicinity of the origin, while keeping the same relative position within the lattice. We call this translation of the vector $\mathbf{v}$ a *lattice modulo reduction.* See Algo. 10.

3.2.8 Orthogonality defect

Many problems in lattice theory have computational solutions proportional to the orthogonality of the basis used to define the lattice. To compare the orthogonality of lattice bases, a metric is required. The orthogonality defect of a basis $\mathbf{B}$ is defined as the product of the vector norms $\|\mathbf{b}_i\|$ in the basis $\mathbf{B}$, divided by the determinant of the lattice $\mathcal{L}_\mathbf{B}$ defined by $\mathbf{B}$.

$$\delta(\mathbf{B}) = \prod_{i=1}^{n} \frac{\|\mathbf{b}_i\|}{det(\mathcal{L}_\mathbf{B})}$$

The orthogonality defect is equal to 1 if and only if the basis $\mathbf{B}$ is completely orthogonal. As orthogonality decreases, the orthogonality defect increases.

It is common to normalize this by taking the n'th root (where n is the rank of the basis $\mathbf{B}$), such that if the vectors are multiplied by some constant factor, the orthogonality defect is scaled by the same factor. [53]

3.3 Lattice Problems

Definition 10 (Shortest Basis Problem (SBP))**.** *Given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, create another basis $\mathbf{B}'$, such that $\mathcal{L}_\mathbf{B} = \mathcal{L}_{\mathbf{B}'}$, where the vectors of $\mathbf{B}'$ are as short as possible for some norm.*

Definition 11 (Approximate Shortest Vector Problem (γ-SVP))**.** *Given a lattice $\mathcal{L}$, the Approximate Shortest Vector Problem is to find a non-zero vector $\mathbf{v} \in \mathcal{L}, \forall \mathbf{u} \in \mathcal{L}, \|\mathbf{v}\| \leq \gamma\|\mathbf{u}\|$.*

Definition 12 (Shortest Independent Vector Problem (SIVP))**.** *Given a basis of a lattice $\mathcal{L}$ and a parameter $q \in \mathbb{Z}$, find the shortest q linearly independent lattice vectors (ie. the set of lattice vectors $\mathbf{b}_1, ..., \mathbf{b}_q$ contained within the minima λ_q).*

Definition 13 (Approximate Closest Vector Problem (γ-CVP))**.** *Let $\mathbf{w}$ be a vector in a lattice $\mathcal{L}$. The Closest Vector Problem is to find a vector $\mathbf{u} \in \mathcal{L}, \forall \mathbf{v} \in \mathcal{L}, \|\mathbf{w} - \mathbf{u}\| \leq \gamma \|\mathbf{v} - \mathbf{u}\|$.*

Definition 14 (Bounded Distance Decoding (BDD))**.** *Given a basis of a lattice $\mathcal{L}$; and a vector $\mathbf{v}$ such that the distance between $\mathbf{v}$ and some $\mathbf{u} \in \mathcal{L}$ is bounded by some parameter; find $\mathbf{u}$. This is a special case of CVP where the vector given is already close to a lattice point.*

3.4 Algorithmic Solutions for Lattice Problems

3.4.1 Gram-Schmidt Orthogonalization

In order to achieve an orthogonal basis, an iterative process can be taken whereby each vector is projected onto a hyperplane perpendicular to the previous vectors. The Gram-Schmidt Orthogonalization algorithm is an iterative approach to orthogonalizing the vectors of a basis. The first vector $\mathbf{b}_1$ of a given basis $\mathbf{B}$ is taken as a reference and the second vector $\mathbf{b}_2$ is projected on to an *(n-1)*-hyperplane perpendicular to $\mathbf{b}_1$. The third vector $\mathbf{b}_3$ is projected onto a *(n-2)*-hyperplane perpendicular to the plane described by $\mathbf{b}_1$ and $\mathbf{b}_2$. This process continues in an iterative fashion until all degrees of freedom are exhausted. The new orthogonal vectors are denoted $\mathbf{b}_i^*$ and the basis as $\mathbf{B}^*$.

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$$

where

$$\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$$

It is important to note however that $\mathcal{L}_{\mathbf{B}^*}$ is not lattice equivalent to $\mathcal{L}_{\mathbf{B}}$ as there need not be any basis of the lattice consisting of orthogonal vectors. $\mathbf{B}^*$ is simply the orthogonal projection of $\mathbf{B}$.

This concept of orthogonalization can be generalized by introducing a function for projecting any vector into orthogonality with the set of basis vectors $b_1, ..., b_{i-1}$:

$$\pi_i(\mathbf{x}) = \sum_{j=i}^{n} \frac{\langle \mathbf{x}, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \mathbf{b}_j^*$$

As a result, $\mathbf{b}_i^*$ can be expressed as $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$.

3.4.2 LLL

The polynomial-time algorithm proposed by Lenstra, Lenstra and Lovaśz[43] (LLL) for increasing the orthogonality of a lattice basis efficiently solves the γ-SVP problem (see Def. 11) for γ exponential in n, by taking any lattice basis as input and computing a short, nearly-orthogonal, lattice-equivalent basis as output, within some bound δ.

We define an LLL-reduced basis as one in which the following two conditions hold:

$$\forall i, j \leq n \quad |\mu_{i,j}| \leq 1/2$$

$$\forall k < n, \quad \delta \|b_k^*\|^2 \leq \|b_{k+1}^*\|^2 + |\mu_{k+1,k}|^2 \times \|b_k^*\|^2$$

To achieve this while maintaining lattice equality, we are only able to use integral row operations (ie. adding and subtracting vectors in the geometric sense).

Through modification of the Gram-Schmidt Orthogonalization algorithm we can achieve this [53].See Algo. 11.

3.4.3 BKZ

The Block Korkine-Zolotarev lattice reduction algorithm [70] can be seen to be a generalization of the LLL algorithm. In particular, the BKZ algorithm effectively performs SVP (using a Korkine-Zolotarev reduction) on blocks (sublattices) of vectors of the original basis and then compares these blocks in a similar way to LLL. If the block-size is 2, BKZ is equivalent to LLL. While larger block-sizes produce bases with a lower orthogonality defect, the running time increases exponentially, limiting practicality. Many improvements can be made to the running time, such as the use of a probabilistic SVP approximation algorithm discovered by Ajtai et al. [8]

3.4.4 Babai's Round-Off

A simple and computationally-fast approximation to the CVP/BDD problem is to express the target vector as a linear, real combination of the basis vectors and round this combination off to integer factors. In effect, this approach finds, for each basis vector, which product of the vector is closer to the target vector. By then adding these products together, the approximate closest vector is found. A geometrical, conceptually simpler explanation is that the target vector is first translated to place it *inside* the fundamental parallelotope, the closest vertex of the fundamental parallelotope is found and this vertex is translated back to it's original position and returned. See Algo. 12.

It is clear however that this approach becomes less effective as the orthogonality of the basis decreases. Since the choice for approximation is limited only to the vertices of the fundamental parallelotope, as the fundamental parallelotope becomes less orthogonal, the target vector may be approximated to a vector that is further away than if we had considered lattice vectors outside the fundamental parallelotope. In fig. 3.1, a basis $\mathbf{B}$ of a 2-dimensional lattice $\mathcal{L}_{\mathbf{B}}$ and an input vector $\mathbf{v}$ is given. Babai's

Round-Off is performed on the input vector and the output **w** is incorrectly computed as the closest vertex of the parallelotope that the vector **v** is inside of. The real closest vector is actually **c**, which is not a vertex of the parallelotope containing **v**.

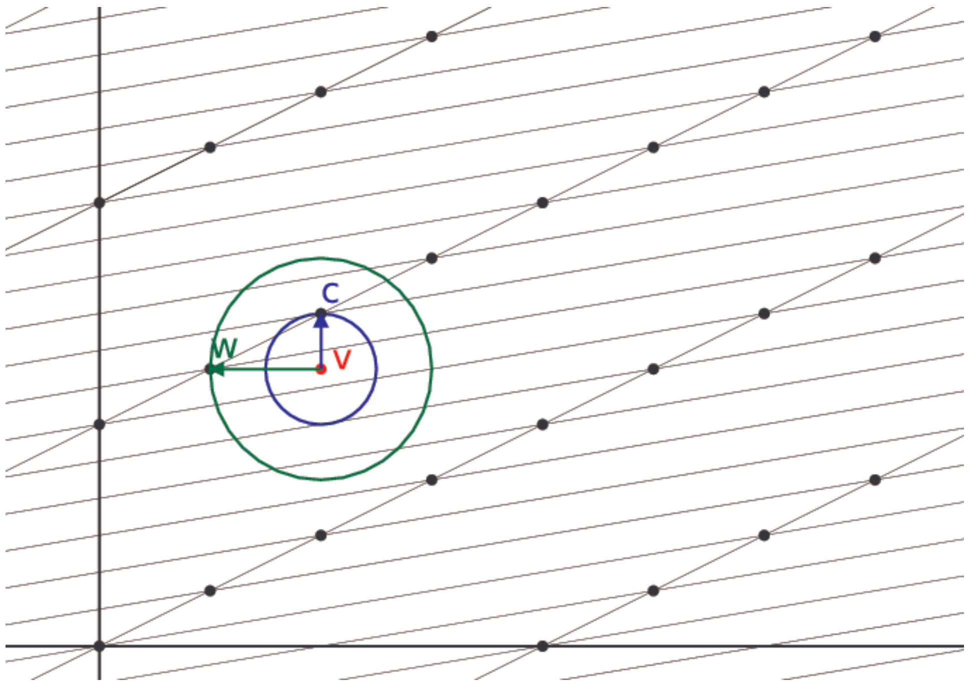

Figure 3.1: Babai's Round-Off algorithm on a low orthogonality lattice basis

With an LLL-reduced basis, the round-off approach finds a close vector with γ-approximation (see [53])

$$\gamma \leq 1 + 2d(9/2)^{d/2}$$

3.4.5 Babai's Nearest-Plane

A better approximation to Babai's Round-Off approach is to consider the hyperplanes of the lattice when assessing the proximity of the target vector. To start, the closest hyperplane of $\{\mathbf{b}_1, ..., \mathbf{b}_{n-1}\}$ to the target vector is found, constrained to steps of $\|\mathbf{b}_n^*\|$. The target vector is then projected on to this hyperplane. These steps are recursively continued for each smaller dimensioned hyperplane until the degrees of freedom drops to one, at which point an approximate closest vector is found. See Algo. 13.

With an LLL-reduced basis, the nearest-plane approach finds a close vector with γ-approximation (see [53])

$$\gamma \leq 2^{d/2}$$

3.5 Lattice Families

3.5.1 *q*-ary Lattices

A special case of lattices in lattice cryptography is that of q-ary lattices. A q-ary lattice $\mathcal{L}$ is defined as that in which any vector which consists of multiples of some scalar q is in the lattice $\mathcal{L}$. ie.

$$q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n, \quad q \in \mathbb{Z}.$$

While any lattice can be represented as a q-ary lattice (for instance if q is a multiple of $det(\mathcal{L})$), primarily q-ary lattice cryptography is concerned with the cases where q is a small, possibly prime value [54]. q-ary lattices have a one-to-one correspondence with linear codes in $\mathbb{Z}_q^n$ [54].

q-ary lattices have seen extensive use recently in producing provably secure lattice-based cryptosystems on q-ary variants of traditional lattice problems. Due to the use of the above special property, lattices which are randomly chosen from a set are able to be used which allows the average-case/worst-case connection as described by Ajtai [5].

3.5.2 Cyclic lattices

Cyclic lattices are a subset of general lattices, where each vector in the lattice basis consists of the coefficients of the previous vector shifted by one position.

By using this construction in the creation of random lattices, enormous gains in efficiency are seen for two reasons. Firstly, storage space decreases linearly as only a vector need be stored as opposed to the entire random matrix, since each row can be derived from a permutation of the first. Secondly, the computation of matrix-vector multiplications can be sped up by using the Fast Fourier Transform if the dimension is a power of two. It is important to note that while the hardness of lattice assumptions in cyclic lattices has not yet been proven, there is no known algorithm for which solving a lattice problem on cyclic lattices is easier than general lattices [54]. The average-case/worst-case connection described by Ajtai [5] also applies to q-ary cyclic lattices.

3.5.3 Ideal lattices

Ideal lattices can be seen to be a superset of cyclic lattices, where instead of a simple shift of co-ordinates distinguishing each lattice vector, a more complex permutation is described. This permutation is described using a vector, which also needs to be stored along with the first vector.

Despite the more complex permutation, the computation and storage efficiency of cyclic lattices is retained to a degree. These lattices are referred to *ideal* lattices as they can be characterized as the *ideals* of a ring of modular polynomials. As with cyclic lattices, there is no known algorithm for which solving a lattice problem on ideal lattices is easier than general lattices [54].

3.6 Cryptosystem Construction using CVP/BDD

In order to produce any asymmetric-key cryptosystem, a hard problem is needed for which an efficient one-way trapdoor can be constructed. If this cryptosystem is to be considered a post-quantum cryptosystem candidate, then the hard problem must be considered hard in the quantum computational sense. In this section, a number of cryptosystems will be discussed, utilizing CVP/BDD as the hard problem and lattice equality as the one way trapdoor.

3.6.1 GGH

In 1996, Goldreich, Goldwasser and Halevi [30] proposed an efficient way to build a cryptosystem that uses lattice theory, inspired by McEliece cryptosystem [49] and based on Bounded Distance Decoding (see Def. 14). Their practical proposition of a cryptosystem was attacked and broken by Nguyen in 1999 [56]. However, the general idea is still viable, as can be seen by the many variants of the basic GGH cryptosystem that have been proposed since (see [26, 52, 61]).

The three general cryptographic algorithms for the GGH class of cryptosystems are as follows:

- **Setup:** Compute a "good basis" $\mathbf{A}$ and a "bad basis" $\mathbf{B}$ of a lattice $\mathcal{L}$. ie.

  $$\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B}).$$

  Provide $\mathbf{B}$ as public and keep $\mathbf{A}$ secret.

- **Encrypt:** To encrypt a plaintext message vector $\mathbf{p}$: Use the bad basis to create a random vector $\mathbf{r}$ of $\mathcal{L}$. Publish the encrypted message which is the addition of the vector message with the random vector: $\mathbf{c} = \mathbf{p} + \mathbf{r}$.

- **Decrypt:** Use the good basis to find the closest vector in the lattice of the encrypted ciphertext message $\mathbf{c}$. The closest vector of the encrypted message $\mathbf{c}$ is the random vector $\mathbf{r}$[1]. Subtract the random vector of the encrypted message obtain the vector message $\mathbf{p} = \mathbf{c} - \mathbf{r}$.

The security of the GGH class of cryptosystems relies on the following three assumptions.

i) It is easy to compute a "bad basis" from a "good basis", but it is difficult to compute a "good basis" from a "bad basis".

ii) It is easy to create a random vector of a lattice even with a "bad basis".

iii) It is easy to find the closest vector with a "good basis" but difficult to do so with a "bad basis".

Goldreich et al. suggested the use of Babai's Round-Off [11] method to address assumption (iii).

After Nguyen's first attack, the utilization of the initial GGH proposition requires lattices with higher dimension (> 500), to ensure security. As a result, the computation of the closest vector, even with a "good basis", becomes very expensive. Another significant drawback to this class of cryptosystems is the enormous key sizes in practical implementations, particularly in such higher dimensions. See Algo. 14 for GGH Key Generation, Algo. 15 for GGH encryption and Algo. 16 for GGH decryption.

3.6.2 The Micciancio Cryptosystem

In 2001, Micciancio [52] proposed some major improvements of the speed and the security of GGH. In this scheme, the public key is of a Hermite Normal Form (HNF) (see Def. 9). Such a HNF basis is not only compact in storage requirements, it also

[1] under the supposition that the norm of $\mathbf{p}$ is sufficiently small

seems to be more difficult to transform to a "good basis" compared to other bases [52]. Furthermore, the HNF of a lattice (and hence the public key of the Micciancio cryptosystem) is unique [18] and can be computed from any lattice basis in polynomial time [38].

The use of such HNF public keys gives space complexity advantages as the resulting public key is much smaller than those proposed by GGH. Rather than Babai's Round-Off method [11] as suggested by Goldreich et al., Micciancio suggested the use of Babai's Nearest-Plane method [11] to address the CVP. However, implementations of this algorithm are extremely slow, again limiting practicality. It is possible however to adapt this cryptosystem to use Babai's Round-Off method and in doing so, much faster decryption speeds are seen, providing that the matrix inverse is precomputed. Unfortunately, the storage requirement for this matrix inverse is extremely large and practicality is again limited. See Algo. 17 for Micciancio Key Generation, Algo. 18 for Micciancio encryption and Algo. 19 for Micciancio Decryption.

3.6.3 Other CVP-based cryptosystems

In 2000, Fischlin and Seifert [26] proposed an original lattice construction with a good basis, with which CVP is easy to solve. In this cryptosystem, the tensor product of the lattice is used to obtain a divide and conquer way to solve CVP.

In 2003, Paeng, Jung and Ha [61] proposed to use a lattice built on polynomial rings. However, in 2007, Han, Kim, and Yeom [35] used a lattice reduction to cryptanalyse this scheme. Their attack recovered the secret key, even in huge dimensions (> 1000) and hence make the PJH scheme unusable.

However, a non broken cryptosystem using polynomial representation exists: NTRU, for N^{th} degree truncated polynomial ring units. NTRU was proposed in 1998 by Hoffstein, Pipher and Silverman [36]. This cryptosystem was not modeled initially as a

GGH type cryptosystem, however, it can been represented as one, which has been useful in analysing the security of the cryptosystem [19].

Chapter 4

Optimisation

Part of this work has been published in QuantumComm 2009 [63] and in ISPEC 2011 [69].

We approached the task of optimising the existing GGH-style cryptosystems in the literature as a modular one. We have identified four aspects of the GGH class of cryptosystems that can be optimised independently to each other. These aspects are:

- Optimisation of Implementation
- Optimisation of Babai's Round-Off in decryption
- Optimisation of the public basis size and encryption speed
- Optimisation of the private key-space

In each of the specialized instances, we will compare our technique with related GGH-style cryptosystems. We will then create a GGH-style cryptosystem created using all techniques and compare it with two related GGH-style cryptosystems.

Since we are primarily concerned with the implementation aspects of the cryptosystems discussed, all of the coding used for the comparisons was performed in C++ using Victor Shoup's NTL [73] (version 5.5.2) compiled against GNU MultiPrecision Library

(GMP) [1] (version 4.3.1). We feel that these libraries are the most appropriate choice for implementation as they are created with runtime speed a major factor in the design while maintaining numerical stability and correctness. With this in mind, the authors feel that this choice of implementation forms a good basis for analysis and comparison. These values were obtained on a 2.1Ghz Intel Core 2 Duo platform with 4Gb RAM. Due to limitations with NTL, only one CPU core was used for testing.

4.1 Optimising Implementation

The theoretical propositions of the CVP-based cryptosystems mentioned in the previous chapter can be well-defined purely in the information theoretic sense, without regard to implementation considerations, in terms of the four required variables; public basis, private basis, plaintext, ciphertext; and the three base operations common to all public-key cryptosystems; key generation, encryption, decryption. A practical implementation based on the information theoretic specification will yield a functionally complete cryptosystem, however the computation time required, particularly in the encrytion and decryption phases, renders such implementations impractical.

4.1.1 Pre-computation

Performance benefits can be seen from moving the computation of some variables from the encryption/decryption phases to the key generation phase. Such variables must be dependent only on the keys and as such can be computed once and stored, to be recalled, rather than computed, when needed.

In the GGH cryptosystem, a precomputation time-space trade-off is specified; The basis inverse is effectively stored in such a way that not only eliminates the need for inverse computation at decryption time, but also removes a vector-matrix multiplication that would otherwise be required. While Micciancio does not specify precomputation of the private basis inverse due to the use of the Nearest-Plane method, by modification of the Micciancio cryptosystem to use the Round-Off method, we see similar performance gains by precomputing the inverse.

Since the round-off step requires the multiplication of a integer vector with a rational matrix (all coefficients having a denominator the determinant), then rounded off, we can simplify this by adding half the determinant to each coefficient, dividing by the determinant and taking the floor of the result. Since the determinant (and indeed half the determinant) can be precomputed, depending upon low-level implementation specifics, this method can be implemented with no decisional branching in the rounding off stage. The benefits of removing decisional branching in the decryption phase is two-fold. Firstly, we see higher performance on pipelined CPU architectures, increasing decryption throughput. Secondly, we can eliminate a source of possible side-channel attacks in the form of branch-prediction analysis. [3]

4.1.2 Chinese Remainder Theorem

Speed increases were noted for some computations specifically involving division of large integers by representing each integer as a sufficient number of residues modulo some prime number. By specifically limiting the primes involved to be strictly smaller than the maximum sized integer representable in a machine word, we are able to perform finite field divison (using the multiplicative inverse) using only integer arithmetic. We can then reconstruct the resulting quotient using the Chinese Remainder Theorem (see Theorem 1). If many separate operations are to be performed, we can significantly speed up the computation by only performing the CRT reconstruction after all operations have been performed. In addition, any such operations operating within the modulo environment are easily adapted to multi-threaded processing platforms.

Theorem 1 (Chinese Remainder Theorem)**.** *Let $m_i \in \mathbb{N}$ be n coprime integers, $M =$*

$\prod_{i=1}^{n} m_i$ *and* $M_i = \dfrac{M}{m_i}$. *Then, for any* n*-tuple* a_i *there exists an unique integer* $0 \leq A < M$ *such that* $a_i = A \bmod m_i$,

$$A = \sum_{i=1}^{n} a_i((M_i)^{-1} \bmod m_i)M_i \bmod M.$$

For a proof of Th. 1 we refer readers to [41].

4.2 Optimising Babai's Round-Off

One method proposed by Babai for solving CVP (and by extension, BDD) within some specified bound is the Round-Off algorithm. While the bound on the resultant solution is not as tight as that of the Nearest-Plane method also specified in Babai's paper, the computational complexity of the Round-Off method is lower, resulting in far greater computation speeds on tested processors. See Figure 4.1 for a comparison of the methods.[1]

A significant speed increase of the Round-Off method can be realised by calculating and storing the private basis matrix inverse as a matrix embedded in a residue number system (RNS). In this way, we are able to perform finite-field multiplication inside the rounding step and reconstruct the original value through the use of the Chinese Remainder Theorem discussed above (See Algo. 1). This is only possible however if an upper bound of the possible values is known.

4.2.1 Improving GGH using the Chinese Remainder Theorem

Due to the use of Babai's Round-Off, the GGH cryptosystem can be optimised using the Chinese Remainder Theorem (CRT) by placing an integer representation of the

[1]This graph shows that practically there is large constant factor between the methods on average. We note that this constant factor does not represent the worst-case ratio. Due to this, we will use the round-off method in our cryptosystem.

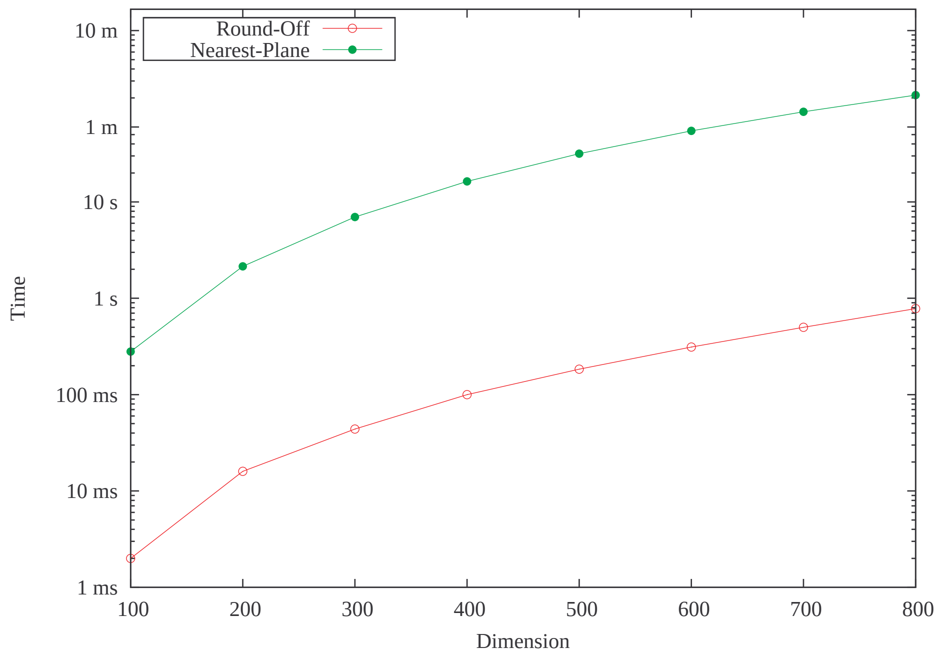

Figure 4.1: Speeds of Round-Off and Nearest-Plane CVP computation

private basis inverse $\mathbf{R}^{-1} \in \mathbb{Q}^{n,n}$ into small finite fields through multiplication with the determinant, then performing the multiplication inside Babai's Round-Off step over these fields. To do this, we first express $\mathbf{R}^{-1}$ as integral by multiplication with $det(\mathbf{R})$ so that we obtain $\mathbf{R}' = \mathbf{R} \times det(\mathbf{R}), \quad \mathbf{R}' \in \mathbb{Z}^{n,n}$. We then calculate the bound of the size of the coefficients. If the coefficients involved include negative coefficients, this bound must be doubled in order to ensure these values are represented uniquely. In random private bases (such as those used by Micciancio), empirical evidence suggests that the number of finite-fields needed is too large to show significant performance gains, if any, over conventional big-integer arithmetic. Conversely, due to the smaller coefficients of the GGH private basis inverse, given c the ciphertext, we define Q as the lower bound of the coefficients in Babai's Round-Off:

$$Q = 2 \times \|\mathbf{c}\|_\infty \times \|\mathbf{R}^{-1}\|_\infty \times det(\mathbf{R})$$

By acknowledging that since GGH specifies the ciphertext as $c = pB + e$,

$$\|\mathbf{c}\|_\infty \leq \|\mathbf{p}\|_\infty \times \|\mathbf{B}\|_\infty + \|\mathbf{e}\|_\infty$$

Thus,

$$Q = 2 \times (\|\mathbf{p}\|_\infty \times \|\mathbf{B}\|_\infty + \|\mathbf{e}\|_\infty) \times \|\mathbf{R}^{-1}\|_\infty \times det(\mathbf{R})$$

Once this bound Q is known, many small finite fields of prime order $m_i < 2^b$ are constructed, where b is the target platforms word size, until the product of these moduli is above the coefficient bound. The decryption round-off calculations are performed independently in these fields and once these have completed, a simple CRT reconstruction is performed to calculate the final value. While the private basis inversion is computationally expensive, we note that if decryption is to be practical, this inversion is assumed to be computed in the key generation phase anyway. As such, the extra computational requirements of the RNS setup is orders of magnitude less significant than the inversion.

4.2.1.1 Construction

Firstly, we define the following scalar function which will be used in the CRT reconstruction.

$$Q(m_i) = \frac{M}{m_i}, \quad M = \prod_{i=1}^{k} m_i$$

Secondly, to avoid rational arithmetic, $\mathbf{R}^{-1} \in \mathbb{Q}^{n,n}$ is multiplied by $det(\mathbf{R})$ to obtain $\mathbf{R}' \in \mathbb{Z}^{n,n}$ ($\mathbf{R}^{-1} = \dfrac{\mathbf{R}'}{det(\mathbf{R})}$). To further save computation at decryption time, we multiply this result with $Q(m_i)$ in the key-generation phase. We therefore define the function $R'(m_i)$ to represent the following:

$$R'(m_i) = \mathbf{R}^{-1} \times det(\mathbf{R}) \times Q(m_i)^{-1} \bmod m_i$$

These functions are precomputed for the set of moduli $\{m_1, m_2, ..., m_k\}$ in the keygeneration phase and stored. Encryption operates as per the original GGH specification however we substitute our CRT Round-Off method discussed above in the decryption stage.

Algorithm 1: CVP Round-Off using the Chinese Remainder Theorem

```
Input  : $\mathbf{v} \in \mathbb{Z}^n$ the input vector, $\mathbf{R} \in \mathbb{Z}^{n,n}$ the private basis,
         $m = \{m_1, m_2, ..., m_q\} \in \mathbb{Z}$ a set of coprime integers, $M = \prod_{i=1}^{q} m_i \in \mathbb{Z}$
         the product of these primes, $M_i = \frac{M}{m_i})^{-1} \bmod m_i \quad \forall i \leq q$
Output: $\mathbf{w} \in \mathcal{L}_\mathbf{R}$ a close vector of $\mathbf{v}$ in the lattice $\mathcal{L}_\mathbf{R}$
begin
    $\mathbf{x} \leftarrow \{0\}^n$
    for $i \leftarrow 1$ to $q$ do
        $\mathbf{x} \leftarrow \mathbf{x} + \frac{M}{m_i} \times (\mathbf{v} \times (\mathbf{R}^{-1} \times det(\mathbf{R})) \times M_i \bmod M$
    end
    $\mathbf{w} \leftarrow \lfloor \mathbf{x}/det(\mathbf{R}) \rceil \times \mathbf{R}$
    return $\mathbf{w}$
end
```

4.2.2 Evaluation of the use of CRT in Babai's Round-Off

We noted large increases in decryption speeds over the general GGH cryptosystem due to the faster multiplication of $\mathbf{cR'}$ inside the RNS (see Fig. 4.2). The number of residues used was strongly correlated with decryption speed and as such we found it necessary to ensure a tight upper bound to prevent using a higher number of residues than required. Small variations were noted, however, and a large number of bases was tested and averaged to obtain these results. As we need to store each matrix in the Residue Number System for $\mathbf{R'}$, we note that the private key size increases slightly (see Fig. 4.3).

We also note that since each finite field is independent, it is possible to decrypt a plaintext via CRT serially with respect to the key, i.e., only loading each matrix over some finite field into memory as we require it. This has great benefits for memory utilization especially for embedded systems as we are only required to store a matrix of standard fixed-precision integers in primary memory at any given time rather than a matrix of much larger variable-precision integers. Obviously in such a case, decryption speeds would be I/O bound in the case of loading each matrix in from a hard drive or flash-based storage.

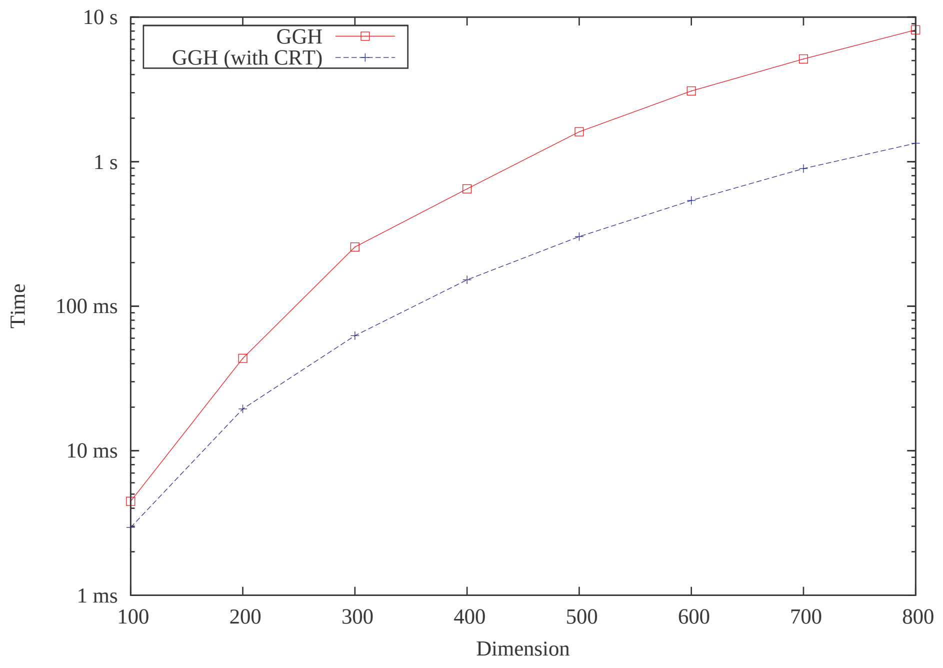

Figure 4.2: Decryption speeds

4.3 Optimising the Public Basis

An analysis of the Micciancio cryptosystem reveals that due to the ciphertext being the modulo reduction of the plaintext by the public basis (see Algo. 10), which itself is of Hermite Normal Form, if this public basis is of a form whereby only one column

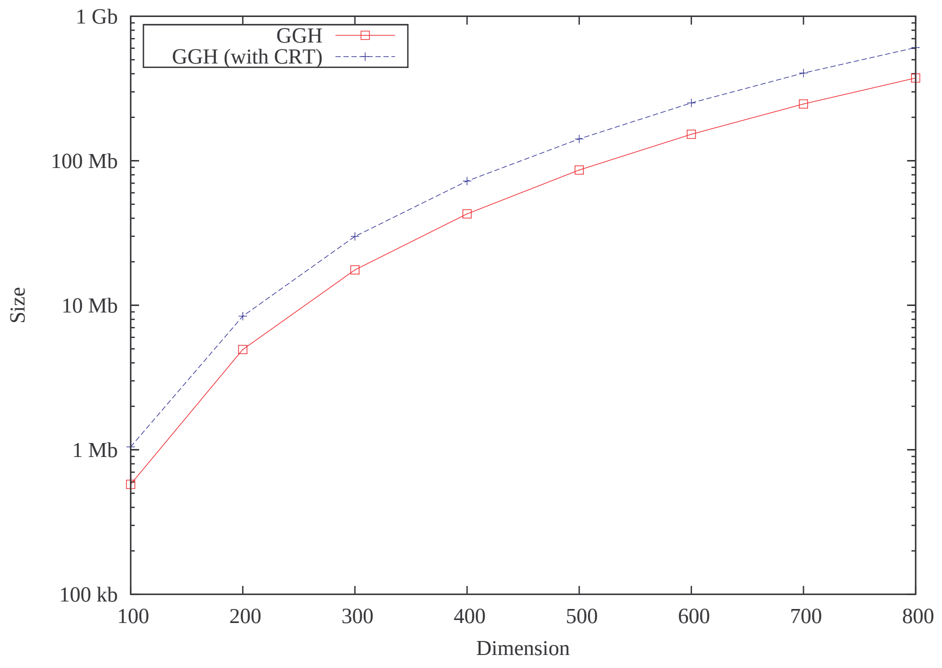

Figure 4.3: Private key sizes

Dimension:	500		800	
	Std.	CRT	Std.	CRT
Encryption Speed (ms)	39.06	38.53	127.8	125.5
Decryption Speed (ms)	1608	302.8	8153	1340
Private Key size (Mbyte)	86.36	141.5	374.2	606.4
Public Key size (Mbyte)	37.80	36.51	149.6	152.6

Table 4.1: Comparison of methods for GGH

is populated, the ciphertext is reduced to a form whereby only one coefficient is non-zero. This allows the ciphertext to be represented as a scalar rather than a vector with no loss of information. While this has benefits in and of itself, such as simplified ciphertext transmission and a reduction in the number of iterations in the encryption and decryption phases, if we modify the decryption method from Babai's Nearest-Plane to Babai's Round-Off, the primary benefits come from the associated storage reduction of the private key. This technique has also been used in Smart and Vercauteren's improvement [75] of Gentry's fully-homomorphic encryption scheme [27] and in the signature scheme developed by Plantard et. al [64].

While it is not strictly required to store the inverse of the private key to perform Babai's Round-Off, doing so dramatically reduces the computation costs of the decryption phase and it is assumed that this inversion occurs in the key generation phase. However, it can be seen that the coefficients of the inverse will be significantly larger than those of the private basis itself, due to the fact that they are not only rational, but also not bound by some arbitrary small number used in the construction. As such, a significant drawback of existing lattice based cryptosystems utilizing such a precomputation step is extremely large storage costs associated with the private key.

Using Babai's Round-Off algorithm [11] rather than the Nearest-Plane method proposed by Micciancio, the first step of ciphertext decryption is to multiply the ciphertext by the inverse of the private basis before rounding off. Since the ciphertext vector is only populated in one position, we are only required to store the corresponding row in the private basis inverse, again without any loss of information. This gives a storage requirement decrease linear in the dimension. In addition, since the time of vector-matrix multiplications is approximately linear in size, we also see a dramatic reduction in decryption times.

We also see faster encryption times as we are able to optimise the vector modulo reduction. Since the ciphertext is now in scalar form and the public basis is in column vector form, we can reduce this step to have an iterative complexity $O(n)$. See Algo. 2.

4.3.1 Naïve testing

A simple way to find such "optimal" Hermite Normal Form bases is to start with a good basis, reduce to Hermite Normal Form and check for optimality.

We define the following:

Algorithm 2: Modulo Reduction of a vector by a lattice with Optimal Hermite Normal Form

```
Input  : $\mathbf{v} \in \mathbb{Z}^n$ the vector to be reduced, $\mathbf{h} \in \mathbb{Z}^n$ the first column vector of a
         lattice basis in Optimal Hermite Normal Form
Output: $b \in \mathbb{Z}$ the modulo reduced scalar
begin
    $b \leftarrow \mathbf{v}_1$
    for $i \leftarrow 2$ to $n$ do
        $b \leftarrow b - (\mathbf{v}_i \times \mathbf{h}_i)$
    end
    $b \leftarrow b \mod \mathbf{h}_1$
    return $b$
end
```

- $Pr[optimal]$ – the probability of the Hermite Normal Form of a random *good* basis being of the aforementioned optimal form
- T_{create} – the time to create a good basis
- T_{HNF} – the time to perform a HNF reduction on a basis and check it is optimal

As such, it can be seen that the mean time taken to generate an optimal hermite normal form via naïve testing is:

$$T_{naïve} = \frac{T_{create} + T_{HNF}}{Pr[optimal]}$$

4.3.2 Coprimality testing

It can be seen that a full-rank basis with a prime lattice determinant and no vectors identical to an identity matrix vector , will produce an optimal HNF, as since the HNF matrix is triangular, the determinant is the product of the diagonal. Furthermore, empirical testing reveals that suboptimal matrices predominantly have low valued diagonals in the non-optimal column, which would not exist had the determinant been coprime with these smaller values. With this in mind, we propose an improved method over the naïve method discussed above. An orthogonal private basis is first created and the determinant calculated. If the determinant is coprime with some set $\mathcal{P}$ of n smallest primes, we can get a much higher probability that the resultant HNF of this matrix is optimal. We specifically define in our implementation that $n = 9$, as the product of the 9 smallest primes is the largest such product that is strictly less than 2^{32} and hence can be represented as an integer on common 32-bit consumer platforms.

We define the following:

- $\mathcal{P} = \{2, 3, 5, 7, 11, 13, 17, 19, 23\}$.
- $Pr[coprime]$ – probability of the determinant of a random good basis being coprime with all elements of the set $\mathcal{P}$.
- T_{check} – the time to calculate the determinant and check coprimality with the set $\mathcal{P}$.

It can be seen that the time to find an optimal basis using this method is the time to create and check coprimality, multiplied by the average number of times this will need to be done in order to achieve determinant coprimality with the set $\mathcal{P}$. The time to calculate the HNF of this basis and check optimality is then added on and the result is multiplied by the average number of times this will need to be done to get optimal, given the co-primality of the determinant:

$$T_{coprimality} = \frac{\left(\frac{T_{create} + T_{check}}{Pr[coprime]}\right) + T_{HNF}}{Pr[optimal|coprime]}.$$

Furthermore, it naturally follows that if we take the limit case for the set $\mathcal{P}$, the zero set (ie. $n = 0$), then we intuitively must define $T_{check} = 0$ and $Pr[coprime] = 1$, leaving us with $T_{na\ddot{i}ve}$:

$$\frac{T_{create} + T_{HNF}}{Pr[optimal]} = T_{na\ddot{i}ve}$$

Empirical testing was performed to ascertain values for $Pr[coprime]$, $Pr[optimal]$ and $Pr[optimal|coprime]$. It is interesting to note that there was no statistically significant deviation in probabilities over the range of dimensions 100 to 1000.

$Pr[optimal]$	0.4345
$Pr[coprime]$	0.0724
$Pr[optimal\|coprime]$	0.9991

Table 4.2: Mean probabilities

Using the probabilities in table 4.2, we are able to compute an average time T_{check} and compare the two aforementioned methods for generating optimal HNF bases.

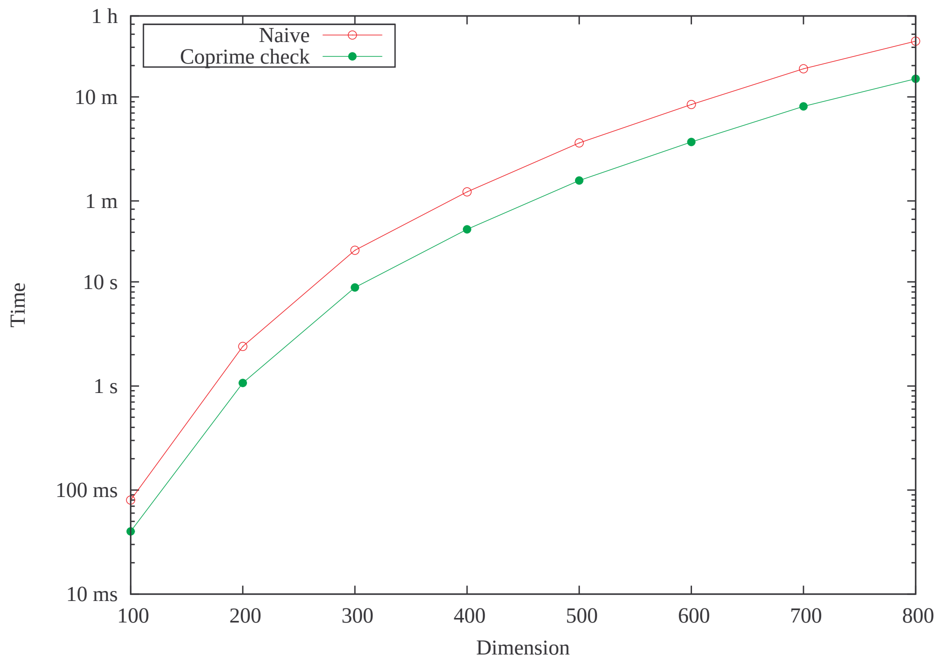

Figure 4.4: Estimated time of optimal HNF basis generation methods

Due to the significant correlation between coprimality and optimality, we found for all dimensions tested that $T_{coprimality}$ was better than twice as fast on average than $T_{na\ddot{i}ve}$. This is due to the extreme time needed to compute the Hermite Normal Form of the basis, which, on average, is optimal with probability less than half.

4.3.3 Evaluation of the use of Optimal Hermite Normal Form Public Bases

For a practical assessment and approximate comparison of keysizes, all keys were compressed with the bzip2 algorithm to mitigate the effect of any redundancies in the data, as the data is not uniform. Decompression times were not included in timing results.

The private key sizes of the standard Micciancio cryptosystem (adapted to use Round-Off decryption) and the variant using Optimal Hermite Normal Form public keys were compared (see Fig. 4.6). As these cryptosystems are compared on both a theoretic and practical basis, we define the private key as also containing redundant, pre-computed private basis inverse. As can be seen, due to the reduced storage requirements of the private basis inverse resulting from the use of Optimal Hermite Normal Forms, our modified GGH cryptosystem has an extremely small private key in comparison to the standard cryptosystem. Similarly, a great improvement in encryption speeds is seen over the standard cryptosystem (See Fig. 4.5.

Dimension:	500		800	
	Std.	OHNF	Std.	OHNF
Encryption Speed (ms)	8.977	0.2315	22.49	0.5546
Decryption Speed (ms)	215.3	208.3	851.4	818.9
Private Key size (Mb)	194.0	0.7405	861.7	2.012
Public Key size (Mb)	0.3925	0.3882	1.085	1.077

Table 4.3: Comparison of methods for Micciancio

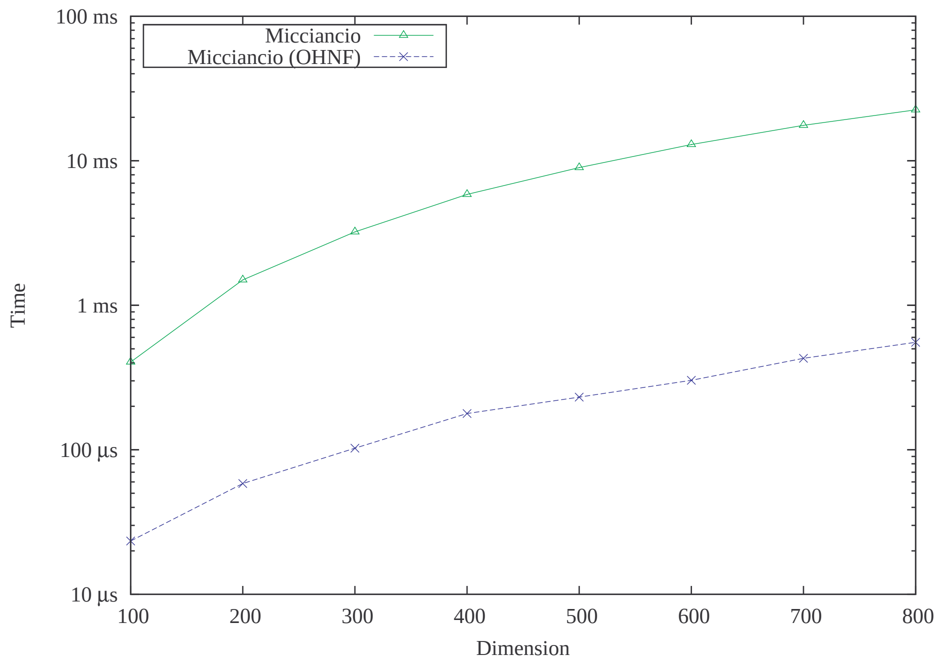

Figure 4.5: Encryption Speeds

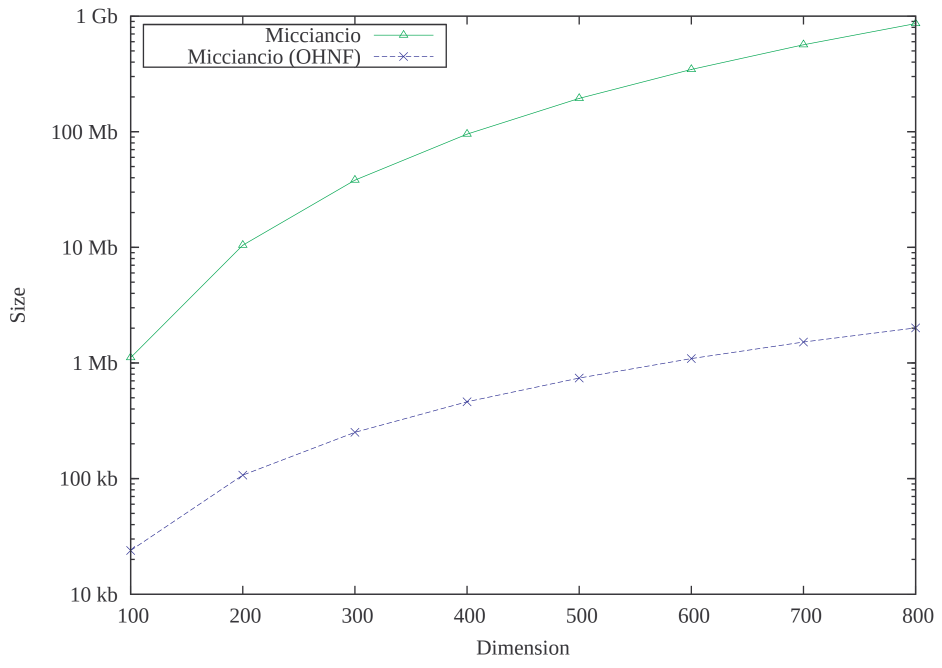

Figure 4.6: Private keysizes

4.4 Optimising the Private Key-Space

4.4.1 Motivation

Both Goldreich et al. and Micciancio specify methods for creating an appropriate lattice basis which has an orthogonality defect sufficient enough to support polynomial time CVP solutions. Traditionally, such a basis was required to be small and nearly-orthogonal, while being sufficiently randomized to prevent basis recovery through CVP-like attacks. Through implementing such cryptosystems, however, a new requirement previously overlooked is that such a basis must be computed with implementation efficiency a priority, in order to be more competitive with traditonal cryptosystems.

While the LLL-reduced [43] private bases proposed by Micciancio can be seen to exhibit good orthogonality, the LLL-reduction step is computationally expensive. While work is constantly being done to improve the computational speed of this reduction (see [57, 71, 55]), we propose a faster method to produce nearly-orthogonal bases, bypassing the LLL step altogether. The private basis construction proposed by GGH has exceptionally good orthogonality as well as being easy and quick to construct, however, the construction restricts the bases to be oriented along the axes. This is unnecessary as the only requirement for CVP is orthogonality and as such, we feel that this poses a limitation on the private keyspace which could possibly be exploited in the future.

Moreover, the particular structure of the GGH basis allows some attacks. For example, instead of looking for some short vector in $\mathcal{L}_{\mathbf{B}}, \mathbf{B} = k\mathbf{I} + \mathbf{M}$, it is easier to look for some close vector of $(k, 0, \ldots, 0)$ in $\mathcal{L}_{\mathbf{B}}$. Since the distance is short ($\|\mathbf{M}_1\|_2$), this limits the security of the private basis. A practical attack will be to look for the shortest vector of the lattice

$$\mathcal{L}_{\mathbf{B}'} = \begin{pmatrix} 0 & B_{1,1} & B_{1,2} & \dots & B_{1,n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & B_{n,1} & B_{n,2} & \dots & B_{n,n} \\ 1 & k & 0 & \dots & 0 \end{pmatrix}$$

which will be easier to find than the shortest vector of $\mathcal{L}_{\mathbf{B}}$. This is due to

$$\lambda_1(\mathcal{L}_{\mathbf{B}'}) < \lambda_1(\mathcal{L}_{\mathbf{B}})$$

To counteract this, we propose a rotation step to be added to the GGH private basis construction phase, which will rotate the lattice probabilistically through a number of planes in the Hilbert Space in which the lattice is embedded, greatly increasing the private key-space and alleviating the aforementioned security issue.

4.4.2 Construction approaches

4.4.2.1 QR-matrices

One method to construct a randomized orthogonal basis non-aligned with the axes is to use the $\mathbf{Q}$ matrix from a $\mathbf{QR}$-decomposition of a matrix $\mathbf{A}$ having uniformly distributed, random coefficients. Since, however, the $\mathbf{R}$ matrix is not needed, yet carries significant information from the original matrix, it is not only wasteful from an information theoretic sense to generate this extra, unnecessary entropy, it also results in a higher computational complexity of $O(n^3)$ [29]. Instead, a method for generating a uniformly distributed randomized $\mathbf{Q}$ matrix directly is sufficient. Several methods exist to create such $\mathbf{Q}$ matrices directly [29], as discussed below.

4.4.2.2 Butterfly orthogonal matrices

The butterfly orthogonal matrices used in Fast Fourier Transforms show promise for two reasons. Firstly, butterfly matrices are computationally inexpensive to generate in fixed precision as they are a product of $\frac{n}{2}$ matrices of complexity $O(1)$. Secondly, the product of a set of butterfly orthogonal matrices is uniformly distributed if the dimension is a power of 2 [29]. Unfortunately, when the dimension is not a power of 2 however, the product of a set of butterfly orthogonal matrices exhibit significant bias with some zero coefficients. These biases can be mitigated through the generation of multiple sets of butterfly orthogonal matrices combined with intermediary permutation matrices which act to distribute the bias, however, this approach requires n permutated butterfly orthogonal matrices to distribute the bias uniformly.

4.4.2.3 Givens-rotation transform matrices

We can generalize the creation of $\mathbf{Q}$ to be a product of some number of independent Givens rotations. To address the shortcoming discussed above, a randomized approach can be taken to the creation of $\mathbf{Q}$ and uniformity can be achieved significantly quicker in practice if the dimension is not a power of 2.

Definition 15 (Givens-rotation transformation matrix)**.** *Let $i, j \in \mathbb{Z}, i \neq j$ represent two distinct axes of a d-dimensional space. Let $\theta \in \mathbb{R}$ be an angle, $-\pi \leq \theta \leq \pi$. A matrix multiplicative transform $G(i, j, \theta)$ is defined as the rotation by θ through the plane defined by the union of the axes i, j, where*

$$\forall p, q \in \mathbb{Z}, 1 \leq p, q \leq d \quad \mathbf{G}_{p,q} \begin{cases} = \sin\theta & \textit{if } p = j \textit{ and } q = i \\ = -\sin\theta & \textit{if } p = i \textit{ and } q = j \\ = \cos\theta & \textit{if } p = q = i \textit{ or } j \\ = 1 & \textit{if } p = q \neq i \textit{ or } j \\ = 0 & \textit{otherwise} \end{cases}$$

This definition can be simplifed by describing the rotation transform as an identity matrix with the following exceptions:

$$\mathbf{G}_{i,i} = \mathbf{G}_{j,j} = \cos\theta$$

$$\mathbf{G}_{j,i} = -\mathbf{G}_{i,j} = \sin\theta$$

An example of such a matrix is as follows:

$$G(2,4,\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

A Givens rotation transform **G** has two properties that allows the construction of a uniformly distributed orthogonal basis. Firstly, since the product of a basis **A** with a Givens rotation transform $G(i, j, \theta)$ is geometrically represented as a rotation through an axial plane defined by the axes (i, j), it follows that the multiplication of any basis with a Givens rotation transform will preserve orthogonality. Secondly, since the identity matrix is trivially orthogonal, the product of an identity matrix with a Givens rotation transform will also be orthogonal, therefore the Givens rotation matrix itself is orthogonal.

By creating a rotation transform $\mathbf{T}$ being the product of a number of independent Givens matrices, choosing each rotation plane and angle randomly, uniformity can be achieved at little computational cost. To ensure uniformity, enough Givens matrices are multiplied to ensure that every coefficient of the resultant basis $\mathbf{T}$ is strictly greater than zero.

To perturb this orthogonal basis and ensure it is integral, the basis is multiplied by some small parameter k and the coefficients are then randomly rounded off to one of the two nearest integers. This can be expressed as $\mathbf{R} \in \mathbb{Z}^{n,n}$ such that $\mathbf{R} = k\mathbf{T} + \mathbf{M}$ where $\mathbf{M} \in \mathbb{R}^{n,n}$. To simplify, this can be represented as $\mathbf{R} = \lfloor k\mathbf{T} \rfloor + \mathbf{M}'$ with $\mathbf{M}' \in [0,1]^{n,n}$. This is a direct, rotated analog of the original GGH specification, alleviating the security issue described above while still maintaining high orthogonality.

4.4.3 Evaluation of Rotated Nearly-Orthogonal Private Bases

While using a rotated nearly-orthogonal basis as a private basis is clearly not as efficient for a given dimension as the original GGH specification due to the necessity of the construction of the rotation matrix, the security parameter using rotated nearly-orthogonal bases is higher due to the larger set of bases it encompasses, even without considering the diagonally-dominant basis attack discussed above.

While it is clear that the original GGH private basis specification key-space is $O(c^{n^2})$ where c is some constant, with n the dimension, it is a more complex proposition to describe the keyspace of a rotated nearly-orthogonal basis, due to the rounding of the real coefficients to integer. As we instantiate the axial-rotation angle θ as a floating-point variable, a naïve approach to measuring the keyspace would result in a keyspace increase exponential in the word-length of the implementation. This is, however, inaccurate as due to the relatively low value of k in the diagonally-dominant pre-rotated basis, the effect of the rotational angle means that the set of all real vectors in the rotated basis may only be rounded to a finite number of integral vectors. The actual domain of a rotated k-length vector is limited to the number of integer vectors less than 1 unit distance from the *(n-1)*-dimension surface of the n-sphere of radius k. It can be seen that this grows exponentially against the dimension n.

4.5 An optimised GGH-style cryptosystem

4.5.1 Overview

We developed a GGH-style cryptosystem incorporating all of the above optimisations and we compared this cryptosystem to both the original GGH specification[2] and Micciancio's improvement. Specifically, this new cryptosystem is defined by having a rotated nearly-orthogonal private basis and an Optimal Hermite Normal Form public basis. Encryption is performed via a lattice modulo reduction as described by Micciancio. Decryption is performed via our improvement to Babai's Round-Off by embedding the large private basis inverse in a Residue Number System.

4.5.1.1 Decryption Error

Due to attacks on lattice cryptosystems with decryption errors (such as Proos' attack on NTRU [65], for a more elaborate discussion see [37]), our scheme has been designed to avoid decryption error through a choice of parameter k and the plaintext domain. Theorem 2 gives a choice of possible parameters where there is no decryption error.[3]

Theorem 2. *Let $k \in \mathbb{R}^+$, $T \in \mathbb{R}^{n,n}$ a rotation matrix, $M \in \mathbb{R}^{n,n}$ with $|M_{i,j}| \leq 1$ and $R = kT + M$. Then for any vectors $c, p \in \mathbb{R}^n$ with $\|p\|_2 < \frac{k-n}{2}$ and $q \in \mathbb{Z}^n$ if $c = p + qR$*

[2] under the assumption that the private key inverse is precomputed in the key-generation phase

[3] In our scheme, we will compute $\|R^{-1}\|_\infty$ and use this to better select the set of parameters

then $q = \lfloor cR^{-1} \rceil$. [4]

Proof of Theorem 2.

<u>Let's study $c - \lfloor cR^{-1} \rceil R$.</u>

$$\begin{aligned} c - \lfloor cR^{-1} \rceil R &= p + qR - \lfloor (p+qR)R^{-1} \rceil R \\ &= p + qR - \lfloor pR^{-1} + q \rceil R \\ &= p - \lfloor pR^{-1} \rceil R \end{aligned}$$

<u>Let's study $\|pR^{-1}\|_\infty$.</u>

$$\begin{aligned} \|pR^{-1}\|_\infty &\le \|pR^{-1}\|_2 \\ &\le \|p\|_2 \|R^{-1}\|_2 \\ &\le \|p\|_2 \|(kT+M)^{-1}\|_2 \\ &\le \|p\|_2 \|(kT)^{-1}(Id + MT^{-1}k^{-1})^{-1}\|_2 \\ &\le \|p\|_2 k^{-1} \|T^{-1}\|_2 \|(Id + MT^{-1}k^{-1})^{-1}\|_2 \end{aligned}$$

<u>Let's study $\|(MT^{-1}k^{-1})\|_2$.</u>

$$\|(MT^{-1}k^{-1})\|_2 \le \|M\|_2 \|T^{-1}\|_2 k^{-1}$$

T^{-1} is a rotation matrix (unitary matrix) then $\|T^{-1}\|_2 = 1$ ([32], Chapter 2.5.6) and $\|M\|_2 \le n \max |M_{i,j}| \le n$ ([32], Chapter 2.3.2, Equation 2.3.8). As by $\frac{k-n}{2} > 0$, we obtain $n^{-1} > k^{-1}$ and

$$\|(MT^{-1}k^{-1})\|_2 < n \times 1 \times n^{-1} = 1.$$

Therefore, as $\|(MT^{-1}k^{-1})\|_2 < 1$, we have $\|(Id + MT^{-1}k^{-1})\|_2 \le \frac{1}{1 - \|MT^{-1}k^{-1}\|_2}$ ([32], Chapter 2.3.4, Lemma 2.3.3). Therefore, we obtain

[4]Therefore, p can be computed correctly from c and B using Babai's Round-Off algorithm.

$$\|pR^{-1}\|_\infty \leq \frac{\|p\|_2 k^{-1}\|T^{-1}\|_2}{1-\|MT^{-1}k^{-1}\|_2} \leq \frac{\|p\|_2\|T^{-1}\|_2}{k-\|M\|_2\|T^{-1}\|_2}$$

Then, $\|pR^{-1}\|_\infty \leq \frac{\|p\|_2}{k-n} < \frac{1}{2}$.Therefore $\lfloor pR^{-1}\rceil = 0$. □

4.5.2 Key Generation

4.5.2.1 Private Key

Rotation matrix: A rotation matrix $\mathbf{T}$ is constructed by taking the product of a sufficient number of randomized givens rotations, such that every coefficient in the product is strictly not equal to 0.

See Algo. 3.

Scaling factor: A noise matrix $\mathbf{N} \in \{0,1\}^{n,n}$ is created and from this a scaling factor k is derived such that $\|(k\mathbf{T}+\mathbf{N})^{-1}\|_\infty < \dfrac{1}{2}$.[5] A naïve method for the computation of this scaling factor k would involve the construction of an intermediary matrix $\mathbf{X} = k\mathbf{T}+\mathbf{N}$, inverting this and taking the infinite-norm. If the infinite norm is found to be above $\dfrac{1}{2}$ then k would be increased and the computation performed again. This would continue until the size of k was sufficient such that $\|\mathbf{X}^{-1}\|_\infty < \dfrac{1}{2}$. While this method would be mathematically correct, matrix inversion is computationally expensive.

Understanding that the determinant of the rotation matrix $\mathbf{T}$ is 1, we can compute the scaling factor k quicker, by understanding that:

$$\|(k\mathbf{T}+\mathbf{N})^{-1}\|_\infty \leq k^{-1} \times (1 + \frac{\|\mathbf{N}\|_\infty}{k} + \frac{\|\mathbf{N}^2\|_\infty}{k^2} + \frac{\|\mathbf{N}^3\|_\infty}{k^3} + ...)$$

We can have a high confidence that the scaling factor is sufficient by taking a

[5]This requirement assures us that there will be no decryption error if $\|\mathbf{p}\|_\infty \leq 1$ as shown in the proof of Theorem 2. By choosing $\|\mathbf{p}\|_\infty \leq 1$, we can minimise the other parameters of the cryptosystem for performance benefits

Algorithm 3: generateRotationTransform(n)

```
Input  : $n \in \mathbb{Z}$ the dimension
Output: $\mathbf{T} \in \mathbb{R}^{n,n}$ the rotation transform matrix
begin
    $\mathbf{T} \leftarrow \mathbf{I}$
    repeat
        $\mathbf{R} \leftarrow \mathbf{I}$
        $i \leftarrow RandInt(1, n-1)$
        $j \leftarrow RandInt(i+1, n)$
        $\theta \leftarrow RandDouble(-\pi, \pi)$
        $\mathbf{R}_{i,i} = cos(\theta)$
        $\mathbf{R}_{j,j} = cos(\theta)$
        $\mathbf{R}_{i,j} = -sin(\theta)$
        $\mathbf{R}_{j,i} = sin(\theta)$
        $\mathbf{T} \leftarrow \mathbf{T} \times \mathbf{R}$
        $anyZeros \leftarrow false$
        for $r \leftarrow 1$ to $n$ do
            for $c \leftarrow 1$ to $n$ do
                if $\mathbf{T}_{r,c} = 0$ then
                    $anyZeros \leftarrow true$
                    $r \leftarrow n$
                    $c \leftarrow n$
                end
            end
        end
    until $anyZeros = false$
    return $\mathbf{T}$
end
```

relatively low number of terms to ensure that the progression is tending towards a value under $\frac{1}{2}$. As each successive term of the progression above decreases the value of the final result, if we choose any fixed number of terms to evaluate, we know that if the result is less than $\frac{1}{2}$, the k chosen is sufficient.

Further performance gains can be seen if we start the iterations with a higher value and increasing it by 1 each iteration until sufficient. In [64] it is conjectured that $k = \left\lceil 2\sqrt{\frac{2n}{3}} \right\rceil$ is sufficient to ensure $\|(k\mathbf{I} + \mathbf{N})^{-1}\| < \frac{1}{2}$ however to be sure, we iterate through k until we can ensure this to be the case.

See Algo. 4.

Algorithm 4: calculateScalingFactor(**N**)

```
Input  : $N \in \{0,1\}^{n,n}$ the noise matrix
Output: $k \in \mathbb{Z}$ the scaling factor
begin
    $k \leftarrow \lceil 2 \times \sqrt{\frac{2\times n}{3}} \rceil$
    $\mathbf{N}' \leftarrow \mathbf{N}$
    $k' \leftarrow k$
    repeat
        $lim \leftarrow 1$
        for $i \leftarrow 1$ to $10$ do
            $lim \leftarrow lim + \frac{\|N'\|}{k'}$
            $\mathbf{N}' \leftarrow \mathbf{N}' \times \mathbf{N}$
            $k' \leftarrow k' \times k$
        end
        $lim \leftarrow \frac{lim}{k}$
    until $\frac{lim}{1-\frac{\|\mathbf{N}'\|}{k'}} < \frac{1}{2}$
end
```

Optimal Hermite Normal Form: The private basis R is constructed:

$$R = \lfloor k\mathbf{T} \rfloor + \mathbf{N}$$

This basis is then checked for coprimality via the method discussed prior (see Algo. 5). If coprimal, **R** is reduced via a HNF decomposition algorithm to produce **B**. **B** is then checked exhaustively to ensure that it is of Optimal Hermite Normal Form (see Algo. 6). If either of these tests fails, the process begins again with a new **N** and k. Since **T** is computationally expensive to generate compared to **N**, we re-use this rotation matrix with the new noise matrix and scaling factor.

Once the public basis is in Optimal Hermite Normal Form, **R** is inverted and multiplied with $det(\mathbf{R})$ to ensure that it is integral. We store this result as an integer row vector $\mathbf{r}' \in \mathbb{Z}^n$, ie. $\mathbf{r}' = (\mathbf{R}^{-1}).row(1) \times det(\mathbf{R})$.

Chinese Remainder Theorem: By modification of the original GGH parameters, we can calculate a bound on the coefficients of $\mathbf{cR}'$ easier than in the original specifi-

Algorithm 5: isCoprime(n, P)

```
Input   : $n \in \mathbb{Z}$ the number to check, $P = \{p_1, p_2, ..., p_k\}$ the set of $k$ primes
Output: $true$ if coprime with the set $P$, $false$ otherwise
begin
    for $i \leftarrow 1$ to $k$ do
        if $n \neq 0 \bmod p_i$ then
            return $false$
        end
    end
    return $true$
end
```

Algorithm 6: isOptimal($\mathbf{A}$)

```
Input   : $\mathbf{A} \in \mathbb{Z}^{n,n}$ the matrix in Hermite Normal Form to check
Output: $true$ if Hermite Normal form is optimal, $false$ otherwise
begin
    for $i \leftarrow 2$ to $n$ do
        if $\mathbf{A}_{i,i} > 1$ then
            return $false$
        end
    end
    return $true$
end
```

cation. Since the public basis $\mathbf{B}$ is of Optimal Hermite Normal Form, we know that $\mathbf{c}$ is strictly less than the lattice determinant. Similarly, as we have defined $\|\mathbf{R}^{-1}\| < 1/2$ we know that $\|\mathbf{R}'\| < \frac{det(\mathbf{R})}{2}$. Since the Residue Number System we're embedding the multiplication in must also handle negative numbers, we need to double this bound.

Therefore,

$$bound = 2 \times det(\mathbf{R}) \times \frac{det(\mathbf{R})}{2} = det(\mathbf{R})^2$$

A number of prime moduli $m_1, m_2, ..., m_q$ are chosen such that the product M of the moduli is greater than the determinant squared. ie.

$$M = \prod_{i=1}^{q} m_i > det(\mathbf{R})^2$$

For optimal computation speeds, we reduce the number of required primes by chosing the highest q primes that can fit in an integer datatype on the target platform. ie. on a 32-bit processor, we choose the highest q primes less than 2^{32}.

We define $M_i = \frac{M}{m_i}$, for each of the moduli m_i being used. This is to reduce unnecessary computations in the decryption phase.

As the public basis is of Optimal Hermite Normal Form, only the first row-vector of $\mathbf{R}'$ is needed. This row-vector is then multiplied by $(M_1)^{-1}$ and reduced modulo m_1. The resultant vector is stored as $\mathbf{r}'_1$. This step is repeated for each modulo m_i up to m_q, resulting in a set $\mathbf{r}'_1, \mathbf{r}'_2, ..., \mathbf{r}'_q$.

$$\mathbf{r}'_i = \mathbf{R}^{-1}.row(1) \times det(\mathbf{R}) \times (M_i)^{-1} \bmod m_i$$

4.5.2.2 Public Key

The public key is the optimal Hermite Normal Form public basis $\mathbf{B}$ of the private basis $\mathbf{R}$. The populated column of $\mathbf{B}$ is stored as a column-vector of length n.

See Algo. 7 for the full key generation algorithm.

Algorithm 7: GenerateKeys(n)

```
Input  : $n \in \mathbb{Z}$ the dimension
Output: $\mathbf{R}, \mathbf{B} \in \mathbb{Z}^{n,n}$ such that $\mathcal{L}_\mathbf{R} \equiv \mathcal{L}_\mathbf{B}$, the array $\mathbf{r'}[i] \in \mathbb{Z}^n$, the array
        $M[q] \in \mathbb{Z}$
begin
    $lowPrimelist \leftarrow$ {2, 3, 5, 7, 11, 13, 17, 19, 23}
    $highPrimelist \leftarrow$ {1073741789, 1073741783, 1073741741, ...}
    $\mathbf{T} \leftarrow$ generateRotationTransform($n$)
    repeat
        repeat
            for $r \leftarrow 1$ to $n$ do
                for $c \leftarrow 1$ to $n$ do
                    $\mathbf{N}_{r,c} \leftarrow$ RandInt(0,1)
                end
            end
            $k \leftarrow$ calculateScalingFactor($\mathbf{N}$)
            $\mathbf{R} \leftarrow \lfloor k\mathbf{T} \rceil + \mathbf{N}$
        until isCoprime($det(\mathbf{R})$, lowPrimeList)
        $\mathbf{B} \leftarrow HNF(\mathbf{R})$
    until isOptimal($\mathbf{B}$)
    $M \leftarrow 1$
    $q \leftarrow 0$
    repeat
        $q \leftarrow q + 1$
        $m_q \leftarrow highPrimeList[q]$
        $M \leftarrow M \times highPrimeList[q]$
    until $M \geq det(\mathbf{R})^2$
    for $i \leftarrow 1$ to $q$ do
        $M_i \leftarrow \frac{M}{m_i}$
        $\mathbf{r'}_i \leftarrow \mathbf{R}^{-1}.row(1) \times det(\mathbf{R}) \times (M_i)^{-1} \bmod m_i$
    end
    $publicKey \leftarrow \mathbf{B}.col(1)$
    $privateKey \leftarrow \{\mathbf{R}, \{\{\mathbf{r'}_1, M_1, m_1\}, \{\mathbf{r'}_2, M_2, m_2\}, ..., \{\mathbf{r'}_q, M_q, m_q\}\}, M\}$
    return $publicKey$, $privateKey$
end
```

4.5.3 Encryption

For encryption, we use Micciancio's method of modulo lattice reduction with the public basis, as this results in a smaller ciphertext. We modified Micciancio's construction by limiting the encryption vector domain to $\{-1, 0, 1\}$ and by using $\|\mathbf{R}^{-1}\|_\infty < 1/2$, we can ensure (See [30]) that there will not be any decryption error (See proof 4.5.1.1). As the public basis is in Optimal Hermite Normal Form, the encryption step can be much faster by reducing the problem to a reduction modulo the column vector. (See Algo. 8). Due to this result, the ciphertext is only populated in the first position and can thus be represented as a scalar. As the remaining columns of the public basis are trivially identical to the identity matrix, the resultant positions in the ciphertext are 0 and do not need to be stored or transmitted. The encryption is not performed under the Residue Number System.

4.5.4 Decryption

Decryption is of a similar form to the improved GGH decryption method using CRT except with a minor change to reflect the encoded message being in the error vector rather than the lattice point. i.e. given the lattice vector $\mathbf{w}$, we calculate the plaintext $\mathbf{p} = \mathbf{c} - \mathbf{w}$. Since the ciphertext is a scalar however, we must either convert this into a vector such that only the first position is populated, or adjust our algorithm to work with the scalar itself. See Algo. 9.

Algorithm 8: Encrypt(m, $publicKey$)

```
Input  : $\mathbf{p} \in \{-1, 0, 1\}^n$ the plaintext, $publicKey = \{\mathbf{b}\}$ where $\mathbf{b} \in \mathbb{Z}^n$ the
         public column vector
Output: $c \in \mathbb{Z}$ the ciphertext scalar
begin
    $c \leftarrow p_1$
    for $i \leftarrow n$ to 2 do
        $c \leftarrow c - (p_i \times \mathbf{b}_i)$
    end
    $c \leftarrow c \bmod \mathbf{b}_1$
    return $c$
end
```

Algorithm 9: Decrypt(c, $privateKey$)

```
Input  : $c \in \mathbb{Z}$ the ciphertext,
         $privateKey = \{\mathbf{R}, \{\{\mathbf{r'}_1, M_1, m_1\}, \{\mathbf{r'}_2, M_2, m_2\}, ..., \{\mathbf{r'}_q, M_q, m_q\}\}, M\}$
         where $\mathbf{R} \in \mathbb{Z}^{n,n}$ the private basis, $\mathbf{r'}_i \in \mathbb{Z}^n$ a private row vector,
         $M_i = \frac{M}{m_i} \in \mathbb{Z}$ a moduli subset product, $m_i \in \mathbb{Z}$ a modulus,
         $M = \sum_{i=1}^{q} m_i \in \mathbb{Z}$ the product of the moduli
Output: $\mathbf{p} \in \mathbb{Z}^n$ the message vector
begin
    $\mathbf{x} \leftarrow 0^n$
    for $i \leftarrow 1$ to $q$ do
        $\mathbf{x} \leftarrow \mathbf{x} + M_i \times (c \times \mathbf{r'}_i \bmod m_i) \bmod M$
    end
    $\mathbf{w} \leftarrow \lfloor \mathbf{x}/det(\mathbf{R}) \rceil \times \mathbf{R} \bmod 2$
    $\mathbf{w}[1] \leftarrow \mathbf{w}[1] - c \bmod 2$
    $\mathbf{p} \leftarrow -\mathbf{w}$
    return $\mathbf{p}$
end
```

4.5.5 Analysis

4.5.5.1 Overview

We compared this cryptosystem with both the GGH cryptosystem using the original parameters and Micciancio's improvement and we saw substantial increases in throughput for both encryption and decryption, together with smaller private and public keys. We used the dimension of these cryptosystems as the basis for comparison as this is the most influential parameter for security.[6] A full comparison between the new cryptosystem defined above and the existing GGH and Micciancio cryptosystems can be seen in Table. 4.4.

Dimension:	500			800		
	GGH	Mic	New	GGH	Mic.	New
Encryption Speed (ms)	39.60	8.977	0.1415	127.8	22.49	0.3155
Decryption Speed (ms)	1608	215.3	140.2	8153	851.4	630.0
Private Key size (Mb)	86.36	194.0	0.6188	374.2	861.7	1.685
Public Key size (Mb)	37.80	0.3925	0.1711	149.6	1.085	0.4654

Table 4.4: Comparison of methods

4.5.5.2 Encryption

We see a substantial increase in encryption speeds over both standard GGH and Micciancio cryptosystems for a number of reasons. Firstly, due to the use of the Optimal Hermite Normal Form public key, we can reduce the iterative complexity of the encryption step by a quadratic factor. Secondly, due to the decreased size of the coefficients in the private basis as a result of using a rotated nearly-orthogonal basis instead of a LLL-reduced random small basis, the public basis coefficients also benefit from a small reduction in size. As a result of this, each arithmetic operation in the encryption phase

[6]We note that in our comparisons, we fixed the length of $\|p\|_\infty \leq 1$ in our tests of the Micciancio cryptosystem as this parameter was undefined in Micciancio's specification.

is reduced in complexity, resulting in further speed gains over the Optimal Hermite Normal Form variant of the Micciancio cryptosystem. See Fig. 4.7.

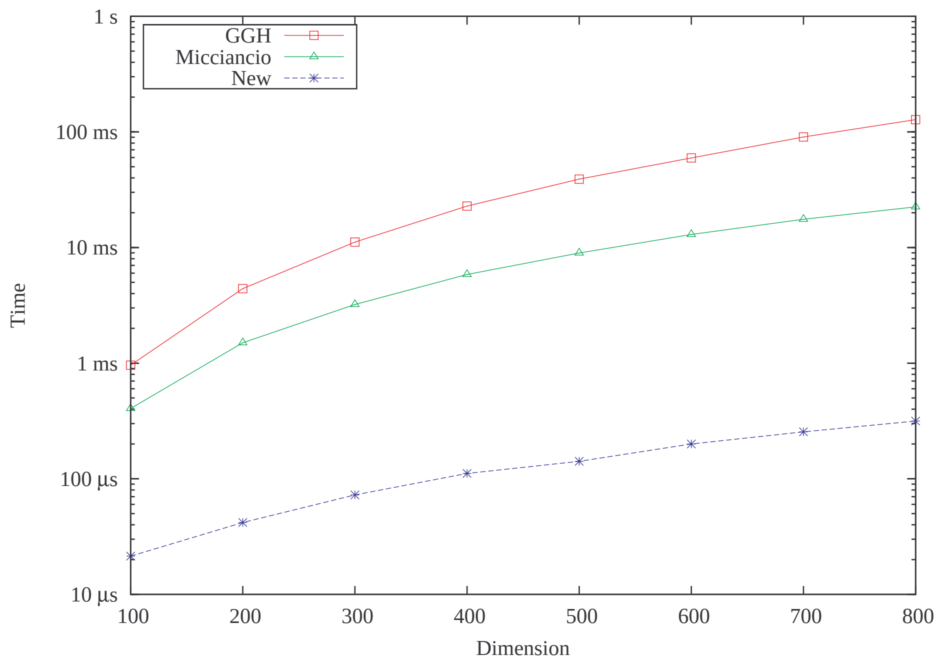

Figure 4.7: Encryption speeds

4.5.5.3 Decryption

We also see improvements to decryption speeds over the existing cryptosystems studied for two reasons. Firstly, by calculating the core multiplication operation over a Residue Number System and the use of the Chinese Remainder Theorem in reconstructing the plaintext, we see a similar factor of improvement to that gained over the standard GGH. In addition to this, because our public key is of Optimal Hermite Normal Form, we are not obliged to store or use any rows of the private basis inverses other than the first, which reduces the complexity of the inner multiplication by a quadratic factor. Lastly, the smaller coefficients of the private basis also result in lower complexity of both the inner core multiplication with the inverse (due to the reduced number of residues and hence iterations required) and the outer vector-matrix multiplication of the rounded vector and the private basis. See Fig. 4.8.

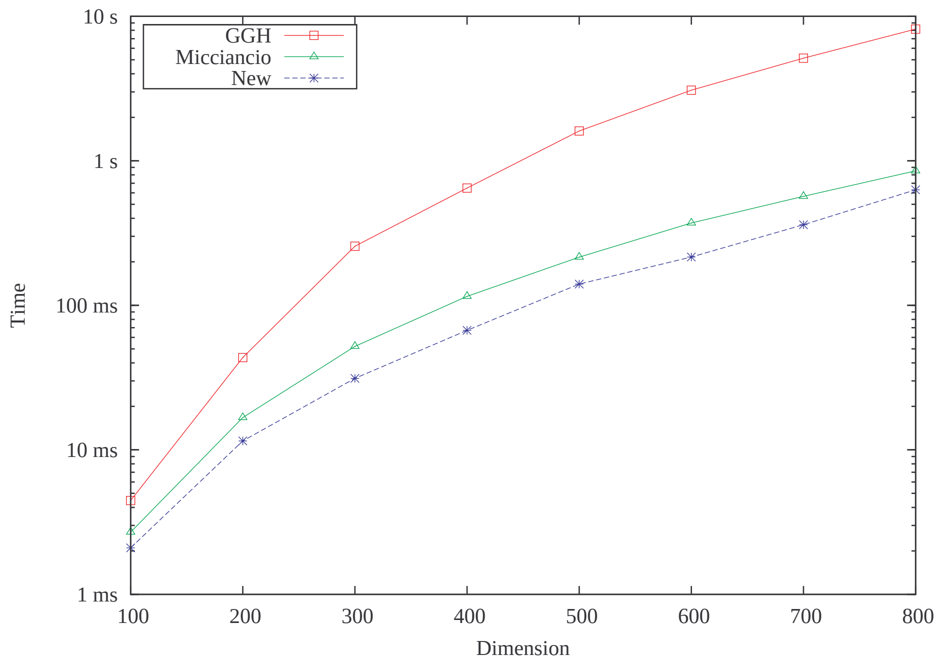

Figure 4.8: Decryption speeds

4.5.5.4 Private Key

The private keys of the new cryptosystem are extremely small compared to the existing GGH and Micciancio cryptosystems and could be considered practical in absolute terms in almost all scenarios. Due to the Optimal Hermite Normal Form public key, only the first row of the private basis inverses need to be stored. As discussed prior, this results in an enormous drop in storage and memory requirements. While the use of the RNS inverse storage increase this storage requirement somewhat, the lower coefficients in the private basis result in smaller coefficients in the private basis inverses, resulting in a decrease in the number of modulo systems required. As these inverses can be placed into the residue number systems at load-time from a ”master” inverse, it could be feasible to store only the master inverse, reducing the storage requirements by a small factor. This avenue of storage saving was not explored however due to the extra load-time complexity and perceived minimal gains. See Fig. 4.9.

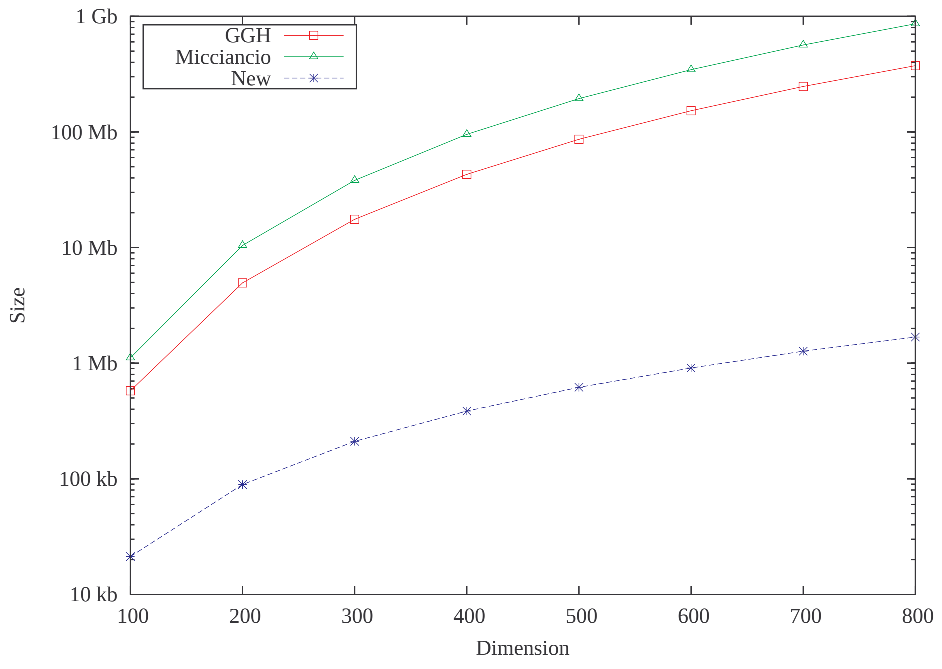

Figure 4.9: Private keysizes

4.5.5.5 Public Key

A reduction in the public key size was seen due to two factors. Firstly, a small gain was seen from the use of the Optimal Hermite Normal Form. Since the non-optimal column in the vast majority of Hermite Normal Form bases are extremely low values, the gains seen here are minimal. Secondly and perhaps more importantly though is the smaller public basis size due to the lower coefficients used in the private basis. Both these factors reduced the public key size to below half that of the Micciancio cryptosystem; and far below GGH at the same dimension. See Fig. 4.10.

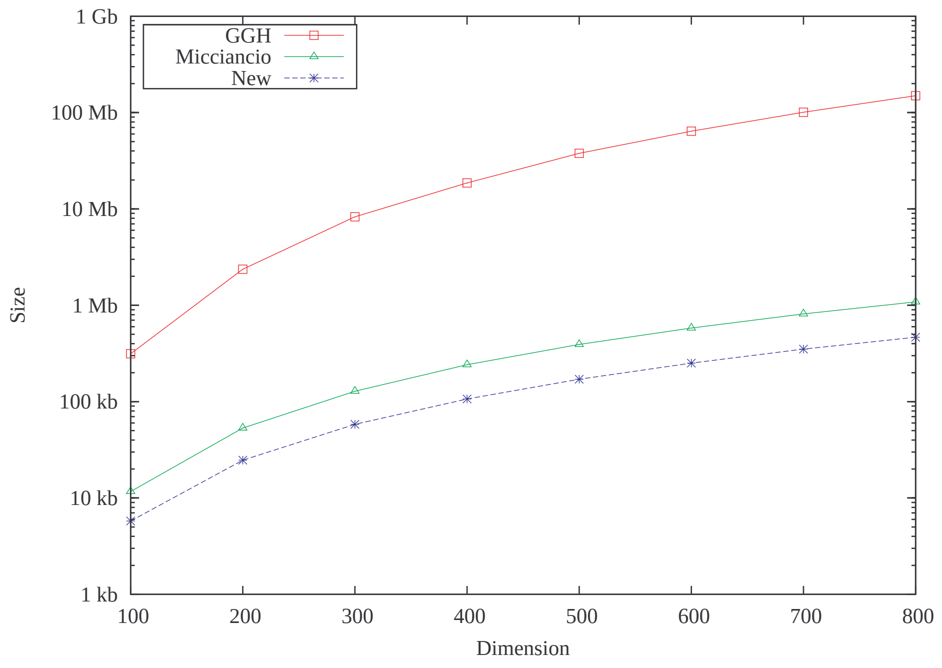

Figure 4.10: Public keysizes

4.5.5.6 Security

Since we have designed this cryptosystem to have no decryption error when the plaintext is in the domain $\{-1, 0, 1\}^n$, we are not obliged to use any mechanism to protect against decryption error attacks (such as [65]). Due to the use of the floor function in both the key generation stage and the decryption (by adding half the determinant to the vector, dividing by the determinant and discarding the remainder) we are also able to mitigate against some side-channel Branch Prediction Analysis attacks as discussed in [3]. We have shown that despite the cryptosystem being vastly more efficient in every metric than the original GGH cryptosystem, the security is enhanced through a much larger private key-space due to the rotated nearly-orthogonal basis. In addition to this larger key-space, we have also shown how some attacks against the diagonally-dominant GGH basis can be prevented.

Chapter 5

Conclusions

The growing urgency for a practical, secure post-quantum cryptosystem has spawned a number of research fields which are working towards this goal. Lattice-based cryptography is showing promise due to a perceived higher security, due to provably NP-hard problems [14], results such as Ajtai's average-case/worst-case equivalence[4] and a large number of provably secure cryptosystems in structured lattice environments. Unfortunately, it remains to be seen whether these structured lattice environments themselves are susceptible to attack. We feel that due to the intense investment of research into cryptosystems operating over these structured lattice environments, it would be prudent to investigate methods for improving general lattice cryptosystems to a level where practical implementation is possible.

The GGH-class[30] of cryptosystems are elegant from a design point of view and appear to offer almost practical keysizes and encryption/decryption speeds. However, the GGH-style cryptosystems in the literature are not competitive with conventional cryptosystems; a prime goal of post-quantum cryptography. We have taken the GGH-class of cryptosystems and have proposed three specific mechanisms to improve the practicality of this class of cryptosystems.

Firstly, we have taken the problem of the inner multiplication in Babai's Round-

Off algorithm that is used in the original GGH specification and sought methods to improve the throughput of this step, as this inner multiplication step is time consuming due to the size of the coefficients involved. We have shown how this multiplication can be placed into a Residue Number System and discussed design considerations that must be taken into account when applying this method. We have analysed the performance gains and have also discussed further optimisations to this process that can be made to increase the speed.

Secondly, we have shown an efficient way to generate a nearly-orthogonal lattice basis such that the Hermite Normal Form basis for that lattice will be non-trivial in only one column. By ensuring that the public basis is in this *Optimal Hermite Normal Form*, we can reduce the size of the private key storage by several orders of magnitude and vastly increase the practicality of GGH-style cryptosystems. As the encryption step described by Micciancio is iterative over the entire public basis, we can also reduce the number of iterations down by an order of magnitude and vastly increase the throughput of this step. Similarly, as decryption time is proportional to the size of the private key, similar performance gains are seen in the decryption stage.

Lastly, we have demonstrated a method for vastly increasing the private keyspace of GGH-style bases by the use of a rotation transform, effectively taking the original GGH specified basis and rotating this through random angles over randomly selected axial-planes. Since the success of Babai's Round-Off algorithm is only dependent upon the basis orthogonality, rotations of the lattice do not affect the efficiency of this step. The rotated semi-orthogonal basis does prevent some attacks on the original GGH basis however, increasing security beyond just the expansion of the private basis keyspace.

Through these techniques, together with some generic optimisations, we have adequately shown how GGH-style cryptosystems can be made practical in many instances, being competitive in terms of encryption and decryption throughput with conventional public-key cryptosystems such as RSA[68]. Similarly, we have demonstrated that we can substantially reduce the size of private keys by an order of magnitude, further improving practical adoption.

Appendix A

Algorithms

A.1 Generic algorithms

Algorithm 10: Modulo reduction of a vector by a lattice

```
Input  : Basis $H \in \mathbb{Z}^{n,n}$ of Hermite Normal Form, $v \in \mathbb{Z}^n$
Output: $b \in \mathbb{Z}^n$
begin
    $b \leftarrow v$
    for $j \leftarrow n$ to 1 do
        $c_j \leftarrow \lfloor b_j \div h_{j,j} \rfloor$
        $b \leftarrow b - c_j \times b_j$
    end
    return $b$
end
```

Algorithm 11: LLL reduction

```
Input  : $B = [b_1, b_2, ..., b_n] \in \mathbb{Z}^{m,n}$
Output: $B \in \mathbb{Z}^{m,n}$, an LLL reduced basis
begin
    loop:
    for $i \leftarrow 1$ to $n$ do
        for $j \leftarrow i-1$ to $1$ do
            $c \leftarrow \left\lfloor \frac{\langle b_i^*, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right\rceil$
            $b_i \leftarrow b_i - c \times b_j$
            for $k \leftarrow 1$ to $n$ do
                if $\delta \|\pi_k(b_k)\|^2 > \|\pi_k(b_{k+1})\|^2$ then
                    Swap($b_k$, $b_{k+1}$)
                    GoTo loop
                else
                    return $B$
                end
            end
        end
    end
end
```

A.2 CVP solutions

Algorithm 12: CVP Round-Off

```
Input  : $v \in \mathbb{Z}^n$ the input vector, $R \in \mathbb{Z}^{n,n}$ a basis of $\mathcal{L} \subseteq \mathbb{Z}^{n,n}$.
Output: $w \in \mathcal{L}$ a close vector of $v$ in the lattice $\mathcal{L}$
begin
    $w \leftarrow \lfloor v \times R^{-1} \rceil \times R$
end
```

Algorithm 13: CVP Nearest-Plane

```
Input  : $v \in \mathbb{Z}^m$ the input vector, $B \in \mathbb{Z}^{m,n}$ a LLL-reduced basis for the lattice
         $\mathcal{L}$.
Output: $w \in \mathcal{L}$ a close vector of $v$ in the lattice $\mathcal{L}$
begin
    $b \leftarrow v$
    for $j \leftarrow n$ to $1$ do
        $c_j \leftarrow \lfloor \langle b, b_j \rangle / \langle b_j, b_j \rangle \rceil$
        $b \leftarrow b - c_j b_j$
    end
    return $t - b$
end
```

A.3 GGH cryptosystem

Algorithm 14: GGH Key generation

Input : $b \in \mathbb{R}$ a number, $n \in \mathbb{Z}$ the security parameter.

Output: $R \in \mathbb{Z}^{n,n}$ a private basis, $B \in \mathbb{Z}^{n,n}$ a public basis, such that $\mathcal{L}_R \equiv \mathcal{L}_B$

RandomNumber(a, b) : Returns one of $\{-1, 0, 1\}$ with probability $\{a, 1-(a+b), b\}$ respectively.

```
begin
    for $i \leftarrow 1$ to $n$ do
        for $j \leftarrow 1$ to $n$ do
            $M_{i,j} \leftarrow$ RandomNumber($\frac{1}{3}, \frac{1}{3}$)
        end
    end
    $R \leftarrow b \times I + M$
    $U \leftarrow I$
    $T \leftarrow I$
    for $i \leftarrow 1$ to $2$ do
        for $c \leftarrow 1$ to $n$ do
            $A \leftarrow I$
            for $r \leftarrow 1$ to $n$ do
                $z \leftarrow$ RandomNumber($\frac{1}{7}, \frac{1}{7}$)
            end
            $A_{c,c} \leftarrow 1$
            $U \leftarrow U \times A$
        end
    end
    $B \leftarrow U \times R$ return $R$, $B$
end
```

Algorithm 15: GGH Encryption

```
Input   : $p \in \mathbb{Z}^n$ a plaintext vector, $B \in \mathbb{Z}^{n,n}$ a public basis.
Output: $c \in \mathbb{Z}^n$ a ciphertext vector
begin
    for $i \leftarrow 1$ to $n$ do
        $q = \frac{\sigma^2}{2 \times \lfloor \sigma \rceil^2}$
        $e_i \leftarrow \sigma \times$RandomNumber($-q, q$)
    end
    $c \leftarrow eB + p$
    return $c$
end
```

Algorithm 16: GGH Decryption

```
Input   : $c \in \mathbb{Z}^n$ a ciphertext vector, $R \in \mathbb{Z}^{n,n}$ a private basis.
Output: $p \in \mathbb{Z}^n$ a plaintext vector
begin
    $p = \lfloor c \times R^{-1} \rceil \times R$
    return $p$
end
```

A.4 Micciancio cryptosystem

Algorithm 17: Micciancio Key Generation

```
Input  : $n \in \mathbb{Z}$ the security parameter.
Output: $R \in \mathbb{Z}^{n,n}$ a private basis, $B \in \mathbb{Z}^{n,n}$ a public basis, such that $\mathcal{L}_R \equiv \mathcal{L}_B$
RandomMatrix(n) : Returns a square matrix of dimension $n$, with random
coefficients uniformly distributed in $\{-n, ..., n\}$.
begin
    $R = LLL(randomMatrix(n))$
    $B = HNF(R)$
    return $R$, $B$
end
```

Algorithm 18: Micciancio Encryption

```
Input  : $p \in \mathbb{Z}^n$ a plaintext vector, $B \in \mathbb{Z}^{n,n}$ a public basis.
Output: $c \in \mathbb{Z}^n$ a ciphertext vector
begin
    $c = p \mod B$
    return $c$
end
```

Algorithm 19: Micciancio Decryption

```
Input  : $c \in \mathbb{Z}^n$ a ciphertext vector, $R \in \mathbb{Z}^{n,n}$ a private basis.
Output: $p \in \mathbb{Z}^n$ a plaintext vector
begin
    $p = \texttt{NearestPlane}(c, R)$ return $p$
end
```

References

[1] The GNU multiple precision arithmetic library.

[2] In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-quantum Cryprography.* Springer, 2008.

[3] Onur Acimez, etin Kaya Ko, and Jean-Pierre Seifert. On the power of simple branch prediction analysis. In *2007 ACM SYMPOSIUM ON INFORMATION, COMPUTER AND COMMUNICATIONS SECURITY (ASIACCS07*, pages 312–320. ACM Press, 2007.

[4] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996)*, pages 99–108, 1996.

[5] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996)*, pages 99–108, 1996.

[6] M. Ajtai. Representing hard lattices with o(n log n) bits. In *STOC*, pages 94–103, 2005.

[7] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Twenty-Ninth Annual ACM Symposium on the Theory of Computing (STOC 1997)*, pages 284–293, 1997.

[8] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 601–610, 2001.

[9] Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFTX: a proposal for the SHA-3 standard, 2008.

[10] D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In *Proceedings of Mycrypt 2005*, volume 3715 of *LNCS*, pages 64–83. Springer, 2005.

[11] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[12] C. H. Bennett and G. Brassard. Quantum cryptography: Public-key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India*, pages 175–179. IEEE Press, 1984.

[13] E. R. Berlekamp. Factoring polynomials over finite fields. *Bell Systems Technical Journal*, 46:1853–1859, 1967.

[14] P. Van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in lattices. Technical Report 81-04, Mathematics Department, University of Amsterdam, 1981.

[15] J. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *Applied Cryptography and Network Security - ANCS 2007*, pages 31–45. Springer, 2007.

[16] J. Buchmann, E. Dahmen, and M. Szydlo. *Post-quantum Cryprography*, chapter Hash-based Digital Signature Schemes, pages 35–93. Springer, 1 edition, 2008.

[17] J.-Y. Cai and T. W. Cusick. A lattice-based public-key cryptosystem. In *Selected Areas in Cryptography*, pages 219–233, 1998.

[18] H. Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics.* Springer-Verlag, 1993.

[19] D. Coppersmith and A. Shamir. Lattice attacks on ntru. In *EUROCRYPT*, pages 52–61, 1997.

[20] D. Coppersmith, J. Stern, and S. Vaudenay. The security of the birational permutation signature schemes. *Journal of Cryptology*, pages 207–221, 1997.

[21] D. Deutsch and R. Jozsa. Rapid solutions of problems by quantum computation. In *Royal Society of London A*, volume 439, page 553, 1992.

[22] J. Ding and B. Y. Yang. *Post-quantum Cryprography*, chapter Multivariate Public Key Cryptography, pages 193–241. Springer, 1 edition, 2008.

[23] C. Dods, N. Smart, and M. Stam. Hash based digital signature schemes. *Cryptography and Coding*, pages 96–115, 2005.

[24] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, June 1982.

[25] J. B. Fischer and J. Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 245–255. Springer-Verlag, 1996.

[26] R. Fischlin and J.-P. Seifert. Tensor-based trapdoors for cvp and their application to public key cryptography. In *IMA Int. Conf.*, pages 244–257, 1999.

[27] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[28] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *STOC*, pages 197–206. ACM, 2008.

[29] A. Genz. Methods for generating random orthogonal matrices. In H. Niederreiter and J. Spanier, editors, *Monte Carlo and Quasi-Monte Carlo Methods*, pages 199–213, 1999.

[30] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(56), 1996.

[31] O. Goldreich, S. Goldwasser, and S. Halevi. Eliminating decryption errors in the ajtai-dwork cryptosystem. In *CRYPTO*, pages 105–111, 1997.

[32] G. H. Golub and C. F. Van Loan. *Matrix Computations, Third Edition.* The Johns Hopkins University Press, 1996.

[33] L. Goubin and N. T. Courtois. Cryptanalysis of the ttm cryptosystem. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 44–57. Springer, 2000.

[34] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 212–219, May 1996.

[35] D. Han, M.-H. Kim, and Y. Yeom. Cryptanalysis of the paeng-jung-ha cryptosystem from pkc 2003. In *Public Key Cryptography*, pages 107–117, 2007.

[36] J. Hoffstein, J. Pipher, and J. H. Silverman. Ntru: A ring-based public key cryptosystem. In *Algorithmic Number Theory (ANTS 1998), Lecture Notes in Computer Science 1423, Springer-Verlag*, pages 267–288, 1998.

[37] N. Howgrave-Graham, P. Q. Nguyen, D. Pointcheval, J. Proos, J. H. Silverman, A. Singer, and W. Whyte. The impact of decryption failures on the security of ntru encryption. In *CRYPTO*, pages 226–246, 2003.

[38] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal of Computing*, 8(4):499–507, 1979.

[39] A. Kawachi, K. Tanaka, and K. Xagawa. Multi-bit cryptosystems based on lattice problems. In *Public Key Cryptography*, pages 315–329, 2007.

[40] A. Kipnis, J. Patarin, and L. Goubin. Unbalanced oil and vinegar signature schemes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 206–222. Springer, 1999.

[41] D. E. Knuth. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1981.

[42] L. Lamport. Constructing digital signatures from a one way function. Technical report, SRI-CSL-98 SRI International Computer Science Laboratory, 1979.

[43] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen, Springer-Verlag*, 261:513–534, 1982.

[44] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Swifft: A modest proposal for fft hashing. In *FSE*, pages 54–72, 2008.

[45] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. Swifft: A modest proposal for fft hashing. In *FSE*, pages 54–72, 2008.

[46] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with error over rings. In *Proceedings of EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2011.

[47] T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature verification and message-encryption. In C. G. Gunther, editor, *Advances in Cryptology - EUROCRYPT 1988*, volume 330 of *LNCS*, pages 419–545. Springer, 1988.

[48] R. McEliece. A public key cryptosystem based on algebraic coding theory. DSN progress report, 1978.

[49] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Deep Space Network Progress Report*, 44:114–116, january 1978.

[50] R. C. Merkle. A certified digital signature. In *Proceedings of Advances in Cryptology - CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.

[51] D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and Lattices Conference (CaLC 2001)*, pages 126–145, 2001.

[52] D. Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and Lattices Conference (CaLC 2001)*, pages 126–145, 2001.

[53] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems, A Cryptographic Perspective.* Kluwer Academic Publishers, 2002.

[54] D. Micciancio and O. Regev. *Post-quantum Cryprography*, chapter Lattice-based Cryptography, pages 147–191. Springer, 1 edition, 2008.

[55] I. Morel, D. Stehlé, and G. Villard. H-lll: using householder inside lll. In *ISSAC*, pages 271–278, 2009.

[56] P. Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from crypto '97. In *Advances in Cryptology - Crypto 1999, Lecture Notes in Computer Science 1666, Springer-Verlag*, pages 288–304, 1999.

[57] P. Q. Nguyen and D. Stehlé. Floating-point LLL revisited. In *Advances in Cryptology - Eurocrypt 2005, Lecture Notes in Computer Science 3494, Springer-Verlag*, pages 215–233, 2005.

[58] P. Q. Nguyen and J. Stern. Cryptanalysis of the ajtai-dwork cryptosystem. In *Advances in Cryptology - CRYPTO 1998*, pages 223–242, 1998.

[59] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control and Inform. Theory*, 15:19–34, 1986.

[60] R. Overbeck and N. Sendrier. *Post-quantum Cryprography*, chapter Code-based Cryptography, pages 95–145. Springer, 1 edition, 2008.

[61] S.-H. Paeng, B. E. Jung, and K.-C. Ha. A lattice based public key cryptosystem using polynomial representations. In *Public Key Cryptography*, pages 292–308, 2003.

[62] J. Patarin. The oil and vinegar signature scheme. Dagstuhl Workshop on Cryptography, September 1997.

[63] T. Plantard, M. Rose, and W. Susilo. Improvement of lattice-based cryptography using CRT. In A. Sergienko, S. Pascazio, and P. Villoresi, editors, *Quantum Communication and Quantum Networking - QUANTUMCOMM 2009*, volume 36, pages 275–282. Springer, 2009.

[64] T. Plantard, W. Susilo, and K. T. Win. A digital signature scheme based on cvp_∞. In R. Cramer, editor, *Proceedings of the 11th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC08)*, volume 4939 of *LNCS*, pages 288–307. Springer Berlin / Heidelberg, 2008.

[65] J. Proos. Imperfect decryption and an attack on the ntru encryption scheme. IACR ePrint Archive, 2003.

[66] O. Regev. Improved inapproximability of lattice and coding problems with preprocessing. In *IEEE Conference on Computational Complexity*, pages 363–370, 2003.

[67] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[68] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb 1978.

[69] M. Rose, T. Plantard, and W. Susilo. Improving BDD cryptosystems in general lattices. In *to appear in Proceedings of ISPEC 2011*, 2011.

[70] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2–3):201–224, 1987.

[71] C.-P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204(1):1–25, 2006.

[72] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring, 1994.

[73] V. Shoup. NTL: Number theory library.

[74] V. Sidelnikov. A public-key cryptosystem based on binary reed-muller codes. *Discrete Mathematics and Applications*, 4(3), 1994.

[75] Nigel Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proceedings of the 13th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC10)*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[76] D. Stehle and R. Steinfeld. Making NTRU as secure as worst-case problem over ideal lattice. In *Proceedings of EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47. Springer, 2011.

[77] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT*, pages 617–635, 2009.