

University of Technology Darmstadt
Department of Computer Science
Cryptography and Computeralgebra

Diploma Thesis

October 2007 - March 2008

Provable Security of Merkle-Micciancio-Lyubashevsky's Signature Scheme



Aleksandar Velikov

University of Technology Darmstadt
Computer Science

Supervised by Prof. Dr. Johannes Buchmann,
Richard Lindner

Acknowledgements

I would like to thank my parents for always believing in me. I would also like to thank my supervisors Richard Lindner and Johannes Buchmann for their useful comments and suggestions on how to improve the quality of the thesis. Not on last place I would like to thank Vadim Lyubashevsky and Luis Carlos Coronado Garcia for their kindness and readiness to answer my questions.

Warranty

I hereby warrant that the content of this thesis is the direct result of my own work and that any use made in it of published or unpublished material is fully and correctly referenced. I also warrant that the presented work has not been used in any other form (even a similar one) in any other examination.

Date: Signature:

Contents

Introduction	12
1 Preliminaries: Definitions and Notations	12
1.1 Types of attacks on digital Signatures	15
1.2 Different kinds to measure the success of the attacker	16
1.3 Expansion factor	16
1.4 Operations for the elements of R^n	17
1.5 Problems in Lattices	18
1.6 Algorithms for finding a solution to the Shortest Vector Problem	18
1.7 Merkle Signature Scheme	20
1.8 Micciancio-Lyubashevsky's hash function	22
1.9 Micciancio-Lyubashevsky's one-time Signature [7]	23
1.10 Notations	23
1.11 The important results from different papers that we will use in this work	24
1.12 Some approximations based on the results from the previous section	30
2 The main result of this thesis	30
2.1 The approximations used in the proof of the main theorem .	36
2.2 What we will see in the next sections	37
3 A useful inequality estimation for the difficulty of finding a collision for the Micciancio-Lyubashevsky's hash	38
4 Security of Micciancio-Lyubashevsky's hash function	42
4.1 Security of the Micciancio-Lyubashevsky's hash function when the attacker use a LLL type reduction algorithm 1.18	42
4.2 Security of the Micciancio-Lyubashevsky's hash function tak- ing into account the results from LLL on the average [42] . .	43
4.3 Security of the Micciancio-Lyubashevsky's hash function (used for hashing the Merkle tree) when the attacker use the Schnorr's RSR algorithm 1.19 (a BKZ lattice reduction algorithm) . . .	45
4.3.1 1 st case: $r = 0$	45
4.3.2 2 nd case : $r > 0$	48
4.4 Security of the Micciancio-Lyubashevsky's hash function (used for hashing the Merkle tree) when the attacker uses the primal- dual RSR algorithm 1.21 (a BKZ lattice reduction algorithm)	50
4.5 Conclusion of this section	52
5 Security of the Micciancio-Lyubashevsky's one-time signa- ture	53

5.1	Security of Micciancio-Lyubashevsky's one-Time Signature [7] when the attacker use a LLL type reduction algorithm 1.18	53
5.2	Security of the Micciancio-Lyubashevsky's one-time signature when we take into account the results from LLL on the average [42]	54
5.3	Security of Micciancio-Lyubashevsky's one-Time Signature [7] when the attacker use the algorithm RSR Schnorr 1.19 (a BKZ lattice reduction algorithm)	55
5.3.1	1 st case: $r = 0$	56
5.3.2	2 nd case: $r > 0$	57
5.4	Security of the Micciancio-Lyubashevsky's one-time signature when the attacker uses the primal-dual RSR algorithm 1.21 (a BKZ lattice reduction algorithm)	59
5.5	Conclusion of this section	61
6	EU-CMA2 Security of MSS (EU-CMA2 = existentially unforgeable under adaptive chosen message attack)	62
6.1	Security estimation without assumptions of the Merkle Signature Scheme against an attacker who uses the primal-dual RSR reduction algorithm 1.21	62
6.2	Security estimation (under assumption 1) of the Merkle Signature Scheme against an attacker who uses the primal-dual RSR reduction algorithm 1.21	64
6.3	Security estimation (under assumption 1) of the Merkle Signature Scheme against an attacker who uses the RSR reduction algorithm	65
6.4	Security estimation (under assumption 1) of the Merkle Signature Scheme against an attacker who uses a LLL-type reduction algorithm	66
6.5	Summary: Simplified Practical Propositions	67
6.5.1	A practical proposition for estimating the security of MLMSS against attacks with primal-dual RSR lattice reduction	67
6.5.2	A practical proposition for estimating the security of MLMSS against attacks with RSR lattice reduction	68
6.5.3	A practical proposition for estimating the security of MLMSS against attacker who uses only LLL-type lattice reductions, but not BKZ-type reduction	69
6.6	Conclusion of this section	69
	Conclusion	70
7	Bibliography	71

A	Appendix section 1	76
A.1	How to turn a collision resistant compression function into a collision resistant hash function	76
A.2	Time complexity of the reduction used for $f - IncSP_{P_\gamma}$ p.10 [4]	76

List of Figures

1	An example of lattice with basis (u, v) in \mathbb{R}^2 with $v = (1, 1)$ and $u = (1, -1)$	13
2	The actual stand of knowledge concerning the Shortest Vector Problem for different values of the parameter γ (see [32]) . . .	20
3	The Merkle Signature Scheme when used with $N = 3$ (and therefore $2^N = 8$ and for the message with number 4	21
4	The plot of the function $\frac{1}{2x} \cdot \log_2\left(\frac{x}{6}\right)$	46
5	The plot of the function $g(x) = x \cdot (3 + \log_2(x))$. In abscissa are the values of k , and in ordinate are the respective values of n . For $n = 1201$, $k = 121$; for $n = 1213$, $k = 122$ and for $n = 2048$ we have $k = 194$	50
6	The plot of the function $g(x) = x \cdot (3 + \log_2(x))$. In abscissa are the values of k , and in ordinate are the respective values of n . For $n = 1951$, $k = 185$; for $n = 1973$, $k = 187$ and for $n = 2048$ we have $k = 194$	59

List of Tables

1	Complexity of the usual lattice algorithms (A table using a part of Richard Lindner’s work [12])	19
2	On average performance of some lattice algorithms	19
3	Security (collision-resistance) of the hash function used for the Signature Tree against LLL-reduction attacks using 1.18	43
4	LLL on average performance and the minimal values of the security parameter n to ensure some security for the Micciancio-Lyubashevsky’s hash function	44
5	Security (collision-resistance) of the hash function against attacks with today’s computer’s power, used for the Signature Tree for different values of the security parameter n against RSR-attacks	48
6	Exact Security (collision-resistance) of the hash function used for the Signature Tree for different values of the security parameter n against RSR-attacks	50
7	Exact Security (collision-resistance) of the hash function used for the Signature Tree for different values of the security parameter n against primal-dual RSR -attacks for $k = 80$	51
8	A resume of section 4	53
9	LLL security of the one-time signature for different values of the security parameter n . One should pay attention to the fact that when choose a polynomial of the form $X^{n-1} \dots + X + 1$ n should be prime and when we choose polynomials of the form $X^n + 1$ n should be a power of 2	54
10	LLL on average security of the one-time signature for different values of the security parameter n	55
11	Exact Security of the one-time signature for different values of the security parameter n against RSR-attacks	59
12	Exact Security of the Micciancio-Lyubashevsky’s one-time signature for different values of the security parameter n against primal-dual RSR -attacks for $k = 80$	60
13	A resume of section 5	62
14	Security of the Merkle Signature Scheme against a primal-dual RSR attack 1.21 for different values of the security parameters n_{crh} (hash function), n_{ots} (one-time signature) and N (2^N number of signatures)	65
15	Security of the Merkle Signature Scheme against a RSR (a BKZ type of attack) for different values of the security parameters n_{crh} (hash function), n_{ots} (one-time signature) and N (2^N number of signatures)	66

16 Security of the Merkle Signature Scheme against a LLL-type attack for different values of the security parameters n_{crh} (hash function), n_{ots} (one-time signature) and N (2^N number of signatures) 67

Introduction

Digital Signatures have gained in importance in recent years. As they provide integrity, authenticity and non-repudiation of data they can be used on the place of handwritten signatures. Moreover they are nowadays in widespread use and therefore their security is of utmost importance. The security of the majority of digital signatures used today such as RSA, DSA, El Gamal and ECDSA is based on the difficulty of factoring large integers and on the discrete logarithm. As Shor demonstrated in [1] quantum computers will make all these signatures insecure. It is not known whether quantum computers that can solve non-trivial factorization problems are feasible but many physicists believe that in the next few decades there will exist powerful enough quantum computers. Up until now only a small quantum computers of seven qu-bits have been created around 2001 which implemented the Shor's algorithm. Unfortunately for the post-quantum age and there are very few alternatives to the traditional signature schemes, every new idea in the area being welcomed with great enthusiasm. Such ideas are for instance to base the security of the signatures schemes on problems in lattices that will remain difficult even with the arrival of quantum computers.

This work deals with a signature scheme proposed by Merkle in his work [2] which we combine with a family of hash functions proposed by Daniele Micciancio and Vadim Lyubachevsky (see [4]) and with a family of one-time signatures proposed by the same authors in [7]. Both the collision resistance of the hash function and the non-forgeability of the one-time signature are based on difficult problems in lattices which we will present later in detail. We prove that the resulting signature scheme is secure and efficient. As there are not known quantum computer algorithms which can solve these lattice problems in polynomial time, the signature scheme is believed to be secure even after the advent of quantum computers i.e. that the Merkle Micciancio-Lyubashevsky Signature Scheme that we propose is post-quantum. For a more detailed study on post-quantum signatures the interested reader can refer to [3].

1 Preliminaries: Definitions and Notations

This section has the goal to introduce some notations and definitions that we will use extensively in the following sections.

Definition 1.1. Let n be a positive integer. A subset \mathcal{L} of the n -dimensional real vector space \mathbb{R}^n is called a lattice if there exists a basis b_1, b_2, \dots, b_n of \mathbb{R}^n , such that:

$$\mathcal{L} = \sum_{i=1}^n \mathbb{Z}b_i = \{ \sum_{i=1}^n r_i b_i : r_i \in \mathbb{Z} \}$$

We say that b_1, b_2, \dots, b_n form a basis of \mathcal{L} (or equivalently that they span \mathcal{L}). We call n the rank of \mathcal{L} . The determinant of the lattice $\det(\mathcal{L})$ is defined as:

$$\det(\mathcal{L}) = |\det(b_1, b_2, \dots, b_n)|$$

The lattice determinant $\det(\mathcal{L})$ is a positive real number that does not depend on the choice of the basis (see proof in [12] p.8 Lemma 1.2).

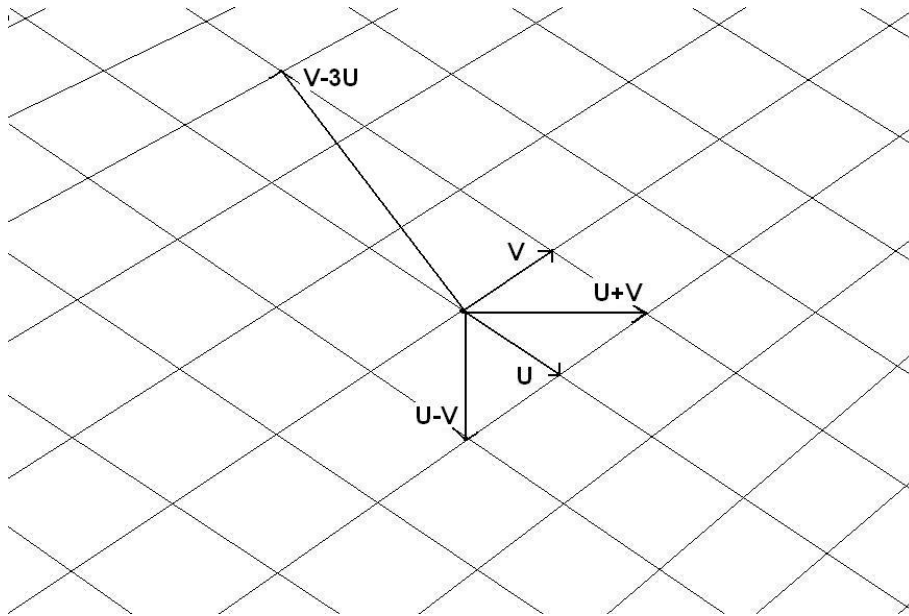


Figure 1: An example of lattice with basis (u, v) in \mathbb{R}^2 with $v = (1, 1)$ and $u = (1, -1)$

Goldwasser, Rivest and Micali introduced some definitions and some concepts of types of attacks on signature schemes [43] :

Definition 1.2. (A Signature Scheme) A signature scheme consists of a triplet of polynomial-time (possibly probabilistic) algorithms (Gen, Sig, Ver) such that for every pair of outputs (sig, ver) of $Gen(1^n)$ and any n -bit message m , $Pr[V(ver, m, S(s, m)) = 1] = 1$ where the probability is taken over the randomness of algorithms Sig and Ver .

Gen is called the key-generation algorithm (it creates a signing key and a verifying key) , Sig is the signing algorithm (it creates signatures of messages with the private key), Ver is called the verification algorithm (it verifies with the public key whether a signature and a message are related through the corresponding private key), and sig and ver are, respectively, the signing and verification keys.

A signature scheme is said to be secure if there is only a negligible probability that any adversary, after seeing signatures of messages of his choosing, can sign a message whose signature he has not already seen. One-time security means that an adversary, after seeing a signature of a single message of his choosing, cannot produce a valid signature of a different message.

Definition 1.3. (A One-Time Signature Scheme) A signature scheme (Gen, Sig, Ver) is said to be one-time secure if for every polynomial-time (possibly randomized) adversary \mathcal{A} , the probability that after seeing $(m, Sig(sig, m))$ for any message m of its choosing, \mathcal{A} can produce $(m' \neq m, \sigma')$ such that $Ver(ver, m', \sigma') = 1$, is negligibly small. The probability is taken over the randomness of Gen, Sig, Ver , and \mathcal{A} .

In the standard security definition of a signature scheme, the adversary should not be able to produce a signature of a message he has not already seen. A stronger notion of security, called strong unforgeability requires that in addition to the above, an adversary should not even be able to come up with a different signature for a message whose signature he has already seen. The scheme presented in this paper satisfies this stronger notion of unforgeability.

We define also the (t, ϵ) -property of a signature scheme in the following way:

Definition 1.4. ((t, ϵ) -property) A scheme or a function or a family of functions have the (t, ϵ) -property \mathcal{P} if no Adversary \mathcal{A} modeled as a probabilistic algorithm which runs within time t , succeeds with probability larger than ϵ in breaking property \mathcal{P} .

Analogously we will define the non (t, ϵ) -property in the following way:

Definition 1.5. (non - (t, ϵ) -property) A scheme or a function or a family of functions have the non - (t, ϵ) -property \mathcal{P} if there exists an Adversary \mathcal{A} modeled as a probabilistic algorithm which runs within time t and which succeeds with probability larger than ϵ in breaking property \mathcal{P} .

The (t, ϵ, n) -existentially unforgeable under adaptive chosen-message attack signature scheme is defined as following:

Definition 1.6. (A (t, ϵ, n) -existentially unforgeable under adaptive chosen-message attack signature scheme) [44] A signature scheme is (t, ϵ, n) -existentially unforgeable under adaptive chosen-message attack, if for all pairs of algorithms $\mathcal{F}_1; \mathcal{F}_2$ running in time at most t :

$$Pr[Ver(F_2(T); PK) = 1 | (SK; PK) \leftarrow Gen(1^s); T \leftarrow \mathcal{F}_1^{Sig(SK; \cdot)}(1^k)] < \epsilon$$

\mathcal{F}_1 requests no more than n signatures from Sig and the message output by \mathcal{F}_2 is different from the messages signed by Sig . The probability is taken over the coin tosses of Gen, Sig, \mathcal{F}_1 and \mathcal{F}_2 . Here $\mathcal{F}_2(T)$ outputs a message/signature.

The previous definition can be used to define the notion of a (t, ϵ) one-time signature scheme:

Definition 1.7. (A (t, ϵ) one-time signature scheme) Let $Sign = (Gen, Sig, Ver)$ be a signature scheme. We say that $Sign$ is a (t, ϵ) one-time signature scheme or *ots* scheme for short, if $Sign$ is $(t, \epsilon, 1)$ existentially unforgeable under adaptive chosen message attack.

1.1 Types of attacks on digital Signatures

Here we propose a review of the different kinds of attacks on Digital Signatures (see [47]). As the security of these signatures is one of the main topics of this work, it would be interesting to see a classification of the possible attacks on a digital signature scheme. There are two main types of attacks: key-only and message attacks:

1. **Key-Only Attacks:** here the attacker knows only the signer's public keys.
2. **Message Attacks:** the attacker is able to examine some signatures corresponding to either known or chosen messages before his attempt to break the system.

We further classify the message attacks in the following categories:

1. **Known Message Attack:** The attacker is given access to a set of signatures for pre-defined messages: m_1, m_2, \dots, m_n . The messages are known to the enemy but are not chosen by him.
2. **Generic Chosen Message Attack:** Here the attacker is allowed to obtain valid signatures for a set of messages of his choice: m_1, m_2, \dots, m_n , before he tries to break the signature scheme. These messages are chosen by the attacker but they are fixed and independent of the public key of the user. This attack is non-adaptive: the entire set of messages is constructed before any signature was seen.
3. **Directed Chosen Message Attack:** Attack similar to the **Generic Chosen Message Attack** with the exception that the set of messages m_1, m_2, \dots, m_n can be created after seeing the signer's public key but before any messages have been created. This attack is still non-adaptive.
4. **Adaptive Chosen Message Attack:** Here the attacker is allowed to use the signer as an "oracle": not only can he request from the signer to sign messages based on the user's public key, but he can also request signatures of messages which depend on previously seen signatures.

The previous attacks are listed in order of increasing severity which means that the enemy's most serious attack is the **Adaptive Chosen Message Attack**. In this work we will consider only this type of attack and we will prove the security of the Micciancio-Lyubashevsky-Merkle signature scheme against this attack.

Moreover in [47] Shafi Goldwasser proposed a classification of the degree of success of a given attack:

1.2 Different kinds to measure the success of the attacker

We consider the following types of success of the attack:

- **Existential Forgery:** Forge a signature for at least one message. The enemy has no control of the message.
- **Selective Forgery:** Forge a signature to a particular message chosen a priori by the enemy.
- **Universal Forgery:** Find an efficient signing algorithm.
- **A Total Break:** Computes the secret trap-door information.

The previous types of breaks are listed in order of increasing severity: the most serious being the **Total Break**.

The Micciancio-Lyubashevsky hash function uses some important notions such as the expansion factor and the ring constant of a polynomial. These notions appear also in the presentation of the Micciancio-Lyubashevsky's theorems (see section 1.11) and we will introduce them in the next subsection:

1.3 Expansion factor

Definition 1.8. (Expansion Factor) The expansion factor of a polynomial $f \in \mathbb{Z}[X]$ for a given integer k is:

$$\text{EF}(f, k) = \max\left\{\frac{\|g\|_f}{\|g\|_\infty} \mid g \in \mathbb{Z}[X], \deg(g) \leq k \cdot (\deg(f) - 1)\right\}$$

Definition 1.9. (The ring constant $\phi(R)$) For any irreducible polynomial of degree n , $f(x)$, we define the ring $R = \mathbb{Z}_p[x]/\langle f \rangle$, where p is some small prime. We can calculate then the quantity $\phi(R)$ as:

$$\phi(R) = \min\{i : \forall a, b \in R, \|ab\|_\infty \leq in\|a\|_\infty\|b\|_\infty\}$$

Remark. In the problems that we will consider we are interested in polynomials with small "expansion factor" and (or) small constant $\phi(R)$.

The following theorems are important because they provide bounds on the expansion factor and the ring constant for some particular types of polynomials. We have the following bounds for the expansion factor (EF) for $x^{n-1} + x^{n-2} + \dots + 1$ and $x^n + 1$ (where n is chosen in such a way that the respective polynomials are irreducible):

Theorem 1.10. [4] $EF(x^{n-1} + x^{n-2} + \dots + 1, k) \leq 2k$, $EF(x^n + 1, k) \leq k$

Remark: The polynomial $x^n + 1$ is irreducible whenever n is a power of 2 and $x^{n-1} + x^{n-2} + \dots + 1$ is irreducible for prime n , so those are good choices for values of n .

For the ring constant $\phi(R)$ we have similar results (again for irreducible polynomials):

Theorem 1.11. [7] $\phi(x^{n-1} + x^{n-2} + \dots + 1) \leq 2$, $\phi(x^n + 1) \leq 1$

1.4 Operations for the elements of R^n

We define the element $\mathbf{a} = (a_1, \dots, a_n) \in R^n$ where $\forall i, a_i \in R$. We define the multiplication of an element in R^n with an element of R in the following way: For $\mathbf{a} \in R^n$ and $y \in R$ we define:

$$\mathbf{a} \cdot y = (a_1 \cdot y, \dots, a_n \cdot y)$$

We also define the following operations in R^n :

• **Scalar Multiplication** For $\mathbf{a} = (a_1, \dots, a_n) \in R^n$ and $\mathbf{b} = (b_1, \dots, b_n) \in R^n$ we define:

$$\mathbf{a} \cdot \mathbf{b} = a_1 \cdot b_1 + \dots + a_n \cdot b_n$$

• **Addition**

$$\mathbf{a} + \mathbf{b} = (a_1 + b_1, \dots, a_n + b_n)$$

Moreover as noted in [7] the set R^n is a R -module because for $\forall \mathbf{a}, \mathbf{b} \in R^n$ and $x, y \in R$ the following properties are satisfied:

- $(\mathbf{a} \cdot x) \cdot y = \mathbf{a} \cdot (x \cdot y)$
- $\mathbf{a} \cdot (x + y) = \mathbf{a} \cdot x + \mathbf{a} \cdot y$
- $(\mathbf{a} + \mathbf{b}) \cdot y = \mathbf{a} \cdot y + \mathbf{b} \cdot y$

Definition 1.12. (Ideal lattice). Let $f \in \mathbb{Z}[X]$ be some monic polynomial of degree n . We define the vector map:

$$\Phi_f : \mathbb{Z}[X]/\langle f \rangle \rightarrow \mathbb{Z}^n : g_0 + \dots + g_{n-1}X^{n-1} \rightarrow (g_0, \dots, g_{n-1})^T$$

A lattice L is ideal if and only if $L = \Phi_f(\Phi_f^{-1}(L) \cdot X)$ for some f .

1.5 Problems in Lattices

In our work we will encounter the following lattice problems:

Definition 1.13. (Shortest Vector Problem, SVP_γ). Given a lattice $\mathcal{L}(B)$, find a nonzero lattice vector Bx (where $x \in \mathbb{Z}^n \setminus 0$) such that $\|Bx\| \leq \gamma \|By\|$ for any $y \in \mathbb{Z}^n \setminus 0$.

Definition 1.14. (Shortest Polynomial Problem, SPP_γ) In the Shortest Polynomial Problem we are given an ideal $I \subseteq \mathbb{Z}^n / \langle f \rangle$ where f is a monic polynomial of degree n , and we must find a $g \in I$ such that $g \neq 0$ and $\|g\|_f \leq \gamma \lambda_1^\infty(I)$.

Definition 1.15. (Incremental Shortest Polynomial Problem, $IncSPP_\gamma(I, g)$) In the approximate Incremental Shortest Polynomial Problem, we are given I and a $g \in I$ such that $\|g\|_f > \gamma \lambda_1^\infty(I)$ and are asked to return an $h \in I$ such that $\|h\|_f \neq 0$ and $\|h\|_f \leq \frac{\|g\|_f}{2}$.

The second and the third problems are far less well studied than the first one, but there are links between these problems for instance a polynomial time reduction from $f - IncSPP_\gamma(I, g)$ to $f - SPP_\gamma(I)$ (see 3.2). Moreover it is conjectured that the best algorithms for solving $f - SPP_\gamma(I)$ are the ones used for SVP_γ (i.e. for instance $f - SPP_\gamma(I)$ is hard to approximate for small values of γ).

1.6 Algorithms for finding a solution to the Shortest Vector Problem

The security of the proposed in this work Merkle - Micciancio-Lyubashevsky Signature Scheme is based on the difficulty to find a solution to the Shortest Vector Problem. The problem of finding an efficient (polynomial time) solution to SVP_γ for lattices in arbitrary dimension was first solved by the celebrated lattice reduction algorithm of Lenstra, Lenstra and Lovasz [9], better known as the LLL algorithm:

Theorem 1.16. (LLL) *There is a polynomial time algorithm to solve SVP_γ for $\gamma = \left(\frac{2}{3}\right)^{\frac{n-1}{2}}$, where n is the rank of the input lattice.*

In [35] Claus Peter Schnorr proposed a parameterized method of reduction of a lattice known as 2k-BKZ, (where BKZ stands for Block-Korkine-Zolotarev) which solves SVP_γ for an approximation factor $\left(\frac{4k}{3}\right)^{\frac{n-1}{2k-1}}$ within time $\mathcal{O}(n^3 k^{k+o(k)} + n^4)$.

There are many variants of the LLL and the BKZ algorithms. The interested reader can see some of them with the corresponding approximation factor and runtime in table 1.

Table 1: Complexity of the usual lattice algorithms (A table using a part of Richard Lindner’s work [12])

Reduction Algorithm	approximation factor $\gamma(n)$	Time
Kannan (83) [21]	1	$2^{\mathcal{O}(n \cdot \log(n))}$
Kumar, Sivakumar (2001) [15]	1	$2^{\mathcal{O}(n \cdot \log(n))}$
Lenstra, Lenstra, Lovasz (82) [9]	$(\frac{4}{3})^{\frac{n-1}{2}}$	$\mathcal{O}(n^5)$
Schnorr (94) [38]	$(\frac{4}{3})^{\frac{n-1}{2}}$	$\mathcal{O}(n^5)$
Koy, Schnorr (2002) [30]	$(\frac{4}{3})^{\frac{n-1}{2}}$	$\mathcal{O}(n^3 \log(n))$
Schnorr (87) [35]	$(\frac{4k}{3})^{\frac{n-1}{2k-1}}$	$\mathcal{O}(n^3 k^{k+o(k)} + n^4)$
Schnorr (03) [37]	$(\frac{k}{6})^{\frac{n}{2k}}$	$\mathcal{O}(n^3 (\frac{k}{6})^{\frac{k}{4}} + n^4)$
Koy (04) [26]	$(\frac{k}{6})^{\frac{n}{k}}$	$\mathcal{O}(n^3 k^{\frac{k}{2}+o(k)} + n^4)$
Blockwise Reduction Revisited [41]	$1.025^{\frac{n}{2}}$	feasible on average, $\mathcal{O}(n^2 \cdot \log(n) \cdot (\frac{k}{11})^{\frac{k}{4}})$

Table 2: On average performance of some lattice algorithms

Performance on average of reduction algorithms	approximation factor $\gamma(n)$ on average	Time
LLL on average ([42])	$1.08^{\frac{n-1}{2}}$	$\mathcal{O}(n^5)$

1

Unfortunately (or fortunately) there are no known efficient algorithms for solving the Shortest Vector Problem in the general case. Moreover SVP_γ is generally known to be NP-hard in its exact ($\gamma = 1$) or even approximate versions for small values of γ , e.g., constant γ independent of the dimension. Therefore no efficient algorithm is likely to exist to solve the problem exactly in arbitrary dimension. For any fixed dimension n , SVP can be solved exactly in polynomial time using an algorithm of Kannan [20]. However, the dependency of the running time on the lattice dimension is $\mathcal{O}(2^{n \log(n)})$. Using randomization and the sieving algorithm of Ajtai, Kumar and Sivakumar [15], exact SVP can be solved probabilistically in $2^{\mathcal{O}(n)}$ time and space.

The collision-resistance of Micciancio and Lyubashevsky’s hash function and the unforgeability of the one-time signature depend on the difficulty of solving an SPP_γ (and therefore through our conjecture also SVP_γ) with $\gamma = \tilde{O}(n)$ and $\gamma = \tilde{O}(n^{2+\epsilon})$ respectively. For a polynomial approximation factor γ Kumar and Sivakumar have found in [10] an algorithm which solves the SVP_γ for an approximation factor $\gamma = n^{3+\epsilon}$ in time $2^{\mathcal{O}(n/\epsilon)}$. There are still no efficient algorithms for solving SVP_γ for a polynomial approximation factor. It is even conjectured that solving SVP_γ within a polynomial factor

¹The result for the time complexity of the Koy’s algorithm is not true for $k \leq 6$ and is only an approximatively true for $k > 6$.

is a hard problem, although not NP-hard (see [22]), since the fastest known algorithms take exponential time ($2^{O(n)}$).

The LLL and BKZ-algorithms have been implemented in many libraries and packages for computational algebra. Among them are:

1. MAGMA Computational Algebraic System (<http://magma.maths.usyd.edu.au/magma/>) - a large, well-supported software package designed to solve computationally hard problems in algebra (probably the best one for LLL at the moment).
2. NTL by Victor Shoup (<http://shoup.net/ntl>) (freely available and it also includes implementation of Block-Korkine-Zolotarev reduction)
3. LiDiA - software package at the TU-Darmstadt (freely available at <http://www.cdc.informatik.tu-darmstadt.de/TI/LiDIA>)
4. PARI - freely available software
5. Maple (<http://www.maplesoft.com/>) - proprietary software
6. Mathematica (<http://www.wolfram.com/products/mathematica/index.html>) - a proprietary software

See the figure for the actual knowledge on the difficulty of solving SVP_γ for different values of the parameter:

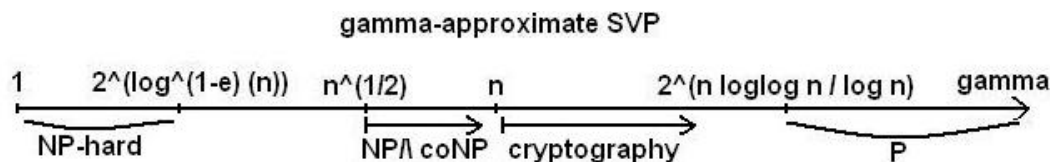


Figure 2: The actual stand of knowledge concerning the Shortest Vector Problem for different values of the parameter γ (see [32])

1.7 Merkle Signature Scheme

Conventional signatures schemes such as RSA and ElGamal do not have any restrictions on the number of signatures allowed to be made during the lifetime of the public and private key. Merkle presented in [2] a multi-time signature scheme which uses the Lamport-Diffie one-time signature scheme. In this work we will use the main idea of Merkle but instead of using Lamport-Diffie one-time signature scheme, we will use the one-time signature scheme proposed by Micciancio and Lubyachevsky in [7]. The main idea of Merkle signature scheme is the following: First of all it assumes that we want to make 2^N signatures.

Merkle Signature Tree For $N = 3$ and for the signature of the message 4

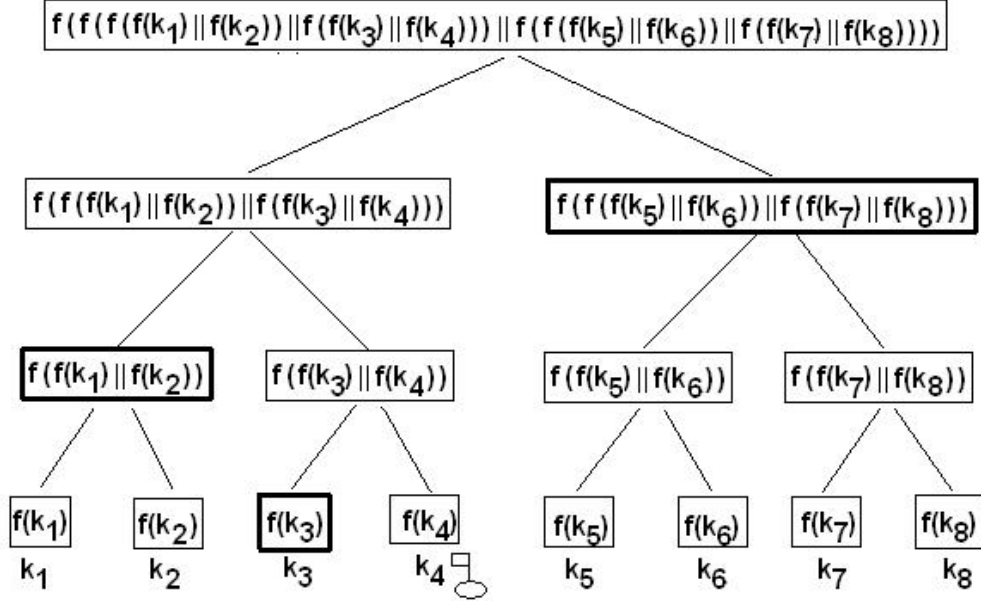


Figure 3: The Merkle Signature Scheme when used with $N = 3$ (and therefore $2^N = 8$ and for the message with number 4

1. *Key Generation:* Generate 2^N key pairs of the one-time signature scheme. After that build a binary tree whose leaves are the hash values of the verification keys, and each parent is the hash value of the concatenation of its left and right children. The private key consists of the one-time key pairs taken together. The public value is the root of the tree.
2. *Signature:* The signature of message number k is the one-time signature with the k^{th} private one-time key, followed by the one-time verification key and N nodes from the tree which help to authenticate the verification key against the public value.

The public key obtained by *Key Generation* is (N, R) . The private key consists of all the one-time key-pairs (X_i, Y_i) . If we can compute efficiently the verifying key Y_i from the signing key X_i then it will be enough to store only the X_i keys.

Moreover the user must keep a counter containing the number j of previously created signatures $0 \leq j \leq 2^N$. At the beginning the counter j is set to zero.

1.8 Micciancio-Lyubashevsky's hash function

(General definition of the Micciancio-Lyubashevsky's hash function [4]) Given a ring $R = \mathbb{Z}_p[x]/\langle f \rangle$, where $f \in \mathbb{Z}[x]$ is a monic, irreducible polynomial of degree n and p is an integer of order roughly n^2 , generate m random elements $a_1, \dots, a_m \in R$, where m is a constant. The ordered m -tuple $h = (a_1, \dots, a_m) \in R^m$ is the hash function. It will map elements in D^m , where D is a subset of R (such that $D = \{g \in R : \|g\|_f \leq d\}$ for some positive integer d), to R . For an element $b = (b_1, \dots, b_m) \in D^m$, the hash is $h(b) = \sum_{i=1}^m a_i \cdot b_i$.

Remark: The size of the key (the hash function) is $O(m \cdot n \cdot \log p) = O(n \cdot \log n)$. The operation $a_i \cdot b_i$ requires time $O(n \cdot \log n \cdot \log \log n)$ using Fourier Transform for appropriate choice of f (see [4]) and as m is constant the hashing requires the same order of time as a function of n .

The hash function introduced in the paper about the Micciancio-Lyubashevsky's hash function (see [4]) is the following:

(Particular Micciancio-Lyubashevsky's hash function used in [4]) In the paper [4] the authors use in particular a hash function family $H(R, D, m)$ composed by instances of the generalized knapsacks (see 1.8) with a polynomial f which has an expansion factor $EF(f, 3) = \mathcal{E}$. The family of functions H is mapping elements as in the general case (see 1.8) from D^m to R where the cardinals of the sets are respectively $|D^m| = (2 \cdot d + 1)^{n \cdot m}$ and $|R| = p^n$. If $m > \frac{\log(p)}{\log^2 d}$, then H will be a family of functions that have collisions.

The hash function introduced in the paper about the Micciancio-Lyubashevsky's one-time signature (see [7]) is the following:

(Particular Micciancio-Lyubashevsky's hash function used in the paper for Micciancio-Lyubashevsky's one-time signature [7]) We define a function family $H(R, m)$ that maps R^m to R . The functions $h \in H(R, m)$ are indexed by elements $\hat{\mathbf{a}} \in R^m$. The input to the function is an element $\hat{\mathbf{z}} \in R^m$, and the output is $\hat{\mathbf{a}} \odot \hat{\mathbf{z}}$ (where the dot product for two elements $\mathbf{a} = (a_1, a_2, \dots, a_m)$ and $\mathbf{z} = (z_1, z_2, \dots, z_m)$ is defined as following $\mathbf{a} \odot \mathbf{z} = (a_1 \cdot z_1, a_2 \cdot z_2, \dots, a_m \cdot z_m)$). So the functions in $H(R, m)$ map elements from R^m to R . To summarize,

$$H(R, m) = \{h_{\hat{\mathbf{a}}} : \hat{\mathbf{a}} \in R^m\}, \text{ where } h_{\hat{\mathbf{a}}}(\hat{\mathbf{z}}) = \hat{\mathbf{a}} \odot \hat{\mathbf{z}}$$

The hash function used for the one-time signature is in fact quite similar to the hash function used in [4] (for the last one we know a theorem for its collision-resistance (see 1.28)). It is mentioned in [7] (page p.6) that the hash function used in [7] also satisfies the collision-resistant theorem 1.28. So for both of the hash functions we know that they are collision resistant.

1.9 Micciancio-Lyubashevsky's one-time Signature [7]

The scheme of the Micciancio-Lyubashevsky's one-time signature consists of three algorithms: **Key-Generation Algorithm**, **Signing Algorithm** and **Verification Algorithm**:

- **Key-Generation Algorithm:**

Input: 1^n , irreducible polynomial $f \in \mathbb{Z}[x]$ of degree n .

Set $p \leftarrow (\phi n)^3$, $m \leftarrow \lceil \log n \rceil$, $R \leftarrow \mathbb{Z}_p[x]/\langle f \rangle$

$\forall i > 0$, let the sets DK_i and DL_i be defined as:

$DK_i = \{\hat{y} \in R^m \text{ such that } \|\hat{y}\|_\infty \leq 5 \cdot i \cdot p^{1/m}, DL_i = \{\hat{y} \in R^m \text{ such that } \|\hat{y}\|_\infty \leq 5 \cdot i \cdot n \cdot \phi \cdot p^{1/m}\}$

Choose uniformly random $h \in \mathcal{H}_{R,m}$.

Choose a positive integer j such that j is chosen with probability 2^{-j}

If $j > \log^2 n$ then set $j \leftarrow \lceil \log^2 n \rceil$.

Pick \hat{k}, \hat{l} independently and uniformly at random from DK_j and DL_j respectively.

Signing Key: (\hat{k}, \hat{l}) . Verification Key: $(h, h(\hat{k}), h(\hat{l}))$

- **Signing Algorithm:**

Input: Message $z \in R$ such that $\|z\|_\infty \leq 1$; signing key (\hat{k}, \hat{l}) .

Output: $\hat{s} \leftarrow \hat{k}z + \hat{l}$

- **Verification Algorithm:** Input: Message z ; signature \hat{s} ; verification key $(h, h(\hat{k}), h(\hat{l}))$

Output: "ACCEPT", if $\|\hat{s}\|_\infty \leq 10\phi p^{1/m} n \log^2 n$ and $h(\hat{s}) = h(\hat{k})z + h(\hat{l})$.

"REJECT", otherwise.

The verification algorithm will always accept the signature generated by the signing algorithm of any message $z \in R$, because the signing keys \hat{k}, \hat{l} are contained in sets $DK_{\log^2 n}$ and $DL_{\log^2 n}$ respectively and thus $\|\hat{k}\|_\infty \leq 5p^{1/m} \log^2 n$ and $\|\hat{l}\|_\infty \leq 5\phi p^{1/m} n \log^2 n$ and therefore,

$$\|\hat{s}\|_\infty = \|\hat{k}z + \hat{l}\|_\infty \leq \phi n \|\hat{k}\|_\infty + \|\hat{l}\|_\infty \leq 10\phi p^{1/m} n \log^2 n$$

Also, by the homomorphic property of functions $h \in \mathcal{H}_{R,m}$,

$$h(\hat{s}) = h(\hat{k}z + \hat{l}) = h(\hat{k})z + h(\hat{l}).$$

1.10 Notations

We will always use the variable n_{crh} as the security parameter of the hash function from 1.8 and the variable n_{ots} for the security parameter of the one-time signature 1.9. In the same way we will always use the index crh for

every problem concerning the security of the hash function (used for hashing the Merkle tree) and we will use the index ots every time we consider the security of the one-time signature.

For instance $\gamma_{\text{crh}} = 8 \cdot \text{EF}(f, 3)^2 \cdot d \cdot m \cdot n_{\text{crh}} \cdot \log^2 n_{\text{crh}}$, and $\gamma_{\text{ots}} = 80 \cdot \phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{\lceil \frac{3}{\log n_{\text{ots}}} \rceil} \cdot \log_2^5(n_{\text{ots}})$. Sometimes when it is clear about which of these two problems we speak we will leave out the index.

We will use the following abbreviations:

- Mss \rightarrow Merkle signature scheme
- MLMSS \rightarrow Micciancio-Lyubashevsky-Merkle signature scheme
- MLots \rightarrow Micciancio-Lyubashevsky's one-time signature
- LLL \rightarrow Lenstra-Lenstra-Lovasz algorithm for lattice reduction
- BKZ \rightarrow Block-Korkine-Zolotarev algorithm for lattice reduction
- GSA \rightarrow Geometric Series Assumption (important assumption in the case of BKZ lattice reduction algorithm RSR proposed by Schnorr in his paper [37])
- RA \rightarrow Randomness Assumption (important assumption in the case of BKZ lattice reduction algorithm RSR proposed by Schnorr in his paper [37])
- Instead of writing "Complexity of a certain problem P " we will write $\mathcal{C}(P)$ (Ex: instead of writing "Complexity of SPP_γ " we will write $\mathcal{C}(SPP_\gamma)$)

1.11 The important results from different papers that we will use in this work

One of the most important results that we will use is the following one from the paper of Kumar and Sivakumar "On polynomial approximations to the shortest lattice vector length" [10] :

Proposition 1.17. *There is a $2^{O(\frac{n}{\epsilon})}$ time algorithm to approximate the length of the shortest vector in an n -dimensional lattice within a factor of $n^{3+\epsilon}$*

In the previous paper the authors argument that even using state of the art lattice reduction methods such as LLL and Block-Korkine-Zolotarev we will not be able to find a more efficient algorithm than the one proposed above for approximation factors of the order of $n^{3+\epsilon}$. LLL finds efficiently an approximation of the shortest vector in a lattice, but this approximation is with an exponential approximation factor. BKZ can not provide us as well

with an efficient algorithm for finding a polynomial approximation factor because its approximation factor is exponential in the reduction parameter k .

Then we will consider the most performant algorithms for lattice reduction:

Theorem 1.18. *(Variant of the LLL-algorithm from Koy, Schnorr (2002) [30]) There is an algorithm which finds an approximation of SVP_γ with $\gamma = (\frac{4}{3})^{\frac{n-1}{2}}$ in time $\mathcal{O}(n^3 \log(n))$.*

Theorem 1.19. *(Variant of the BKZ-algorithm (RSR) from Schnorr (2003) [37]) There is an algorithm which finds an approximation of SVP_γ with $(\frac{k}{6})^{\frac{n}{2k}}$ in time $\mathcal{O}(n^3(\frac{k}{6})^{\frac{k}{4}} + n^4)$.*

Remark: In the RSR algorithm the following inequalities $k \geq 24$, $3 \cdot k + k \cdot \log_2(k) \leq n$ should be satisfied as well as two assumptions: Geometric Series Assumption (GSA), Randomness Assumption (RA) ([37]).

Theorem 1.20. *(Variant of the BKZ-algorithm (primal-dual) from Koy (2004) [26]) There is an algorithm which finds an approximation of SVP_γ with $(\frac{k}{6})^{\frac{n}{k}}$ in time $\mathcal{O}(n^3 k^{\frac{k}{2} + o(k)} + n^4)$.*

Remark: In the Koy's primal-dual algorithm the assumptions Geometric Series Assumption (GSA) should be satisfied.

Theorem 1.21. *(Primal-dual RSR, $k = 80$ by heuristic, RA, GSA [41]) There is an algorithm which finds an approximation of SVP_γ with $\gamma = (1.025)^{\frac{n}{2}}$ in time feasible in practise.*

Remark: In the Primal-dual RSR, $k = 80$ algorithm the assumptions Geometric Series Assumption (GSA) and Randomness Assumption (RA) should be satisfied (see [41]).

We will use the above 4 algorithms: among them one is of the LLL type (the most efficient known of this type), and the others are of the BKZ type. All of the BKZ type algorithms make the assumption GSA which is not satisfied in most cases [12].

The crucial theorem that we will use for the estimation of the security of the Micciancio-Lyubashevsky-Merkle signature scheme is due to Johannes Buchmann and Richard Lindner and inspired by the dissertation of Luis Carlos Coronado Garcia (see Proposition 1.5 p.31 [25]) :

Theorem 1.22. *(MSS security) Let H be a s -bit, (t_{cr}, ϵ_{cr}) collision resistant family of hash-functions parametrized by \mathcal{K} (we will use the hash function from [4]), and let OTS be a $(t_{ots}, \epsilon_{ots})$ one-time signature.*

Then for any number N we may set:

$$t := \max\{t_{cr} - (N + 1) \cdot t_H, t_{ots}\} - t_G - 2^N \cdot t_S$$

$$\epsilon := 2\max\{\epsilon_{cr}, 2^N \cdot \epsilon_{ots}\}$$

and if $t > 0$ and $\epsilon \leq 1$ the original Merkle signature is $(t, \epsilon, 2^N)$ existentially unforgeable under adaptive chosen message attacks. Here t_G is the time taken to generate an Mss instance, t_S is the time taken for signing a message using Mss, and t_H is the time taken for hashing.

Proof Δ We show the theorem by the contrapositive: We assume that $Pr(\text{"successful forgery of Mss"}) \geq \epsilon$ in time t and show that either $Pr(\text{"finding collision for [4]"}) \geq \epsilon_{cr}$ in time t_{cr} or $Pr(\text{"finding successful forgery for ots"}) \geq \epsilon_{ots}$ in time t_{ots} .

To do this we assume the existence of forging algorithm \mathcal{F} which either takes an Merkle signature scheme public key $(\mathcal{K}, \mathcal{R})$ and lets us sign a 2^N query messages M_i for $0 \leq i \leq 2^N - 1$ to produce a valid forgery (M', σ') or takes a Merkle ots public key (K, \mathcal{Y}) and let us sign a single query message M to produce a valid forgery for MLots (M', τ') .

Using \mathcal{F} we show the existence of an algorithm \mathcal{A} which given a public key \mathcal{Y} of the underlying ots and a corresponding oracle \mathcal{O}_{ots} , will either find a collision for [4] or a forgery for the ots [7].

Our \mathcal{A} works as follows We start by selecting a hash parameter K uniformly at random. We also randomly select an index $0 \leq i_0 \leq 2^N - 1$. Then we generate an Mss instance, but during the generation we replace the leaf with index i_0 with the hash of the public key given to us $L_{N,i_0} = H_K(\mathcal{Y})$. The public key $(\mathcal{K}, \mathcal{R})$ of this Mss instance is transferred to \mathcal{F} .

We can sign all 2^N query messages that \mathcal{F} may ask us by using the oracle for the message with index i_0 . So we obtain a forgery (M', σ') from \mathcal{F} . Let the index of the forgery be i . We know that we provided \mathcal{F} with the signature (M_i, σ_i) . We distinguish between two distinct cases and let the probability of 1st Case be ρ :

1st case - $(Y_i, P_N, \dots, P_1) \neq (Y', P'_N, \dots, P'_1)$: In this case we first check whether $Y_i \neq Y'$ and $H_K(Y_i) = H_K(Y')$, if it is the case, we have produced a collision because we have two different leaves or sibling path with respect to the same root R , and they both are in a hash tree at the same position i . To actually produce the collision values, we need to hash at most $N - 1$ times. For a detailed proof of this result the interested reader can see Lemma 1.4 p.30 [25].

2nd case - $(Y_i, P_N, \dots, P_1) = (Y', P'_N, \dots, P'_1)$: In this case we need the additional condition that $i = i_0$ which happens with probability 2^{-N} . If indeed $i = i_0$ we have produced a forgery for the ots, because there has to be a difference in the remaining part of the Mss signature. So we have $(M_{i_0}, \tau_{i_0}) \neq (M', \tau')$ and these are both valid signatures with respect to the ots public key $Y = Y_{i_0}$ which we wanted to produce a forgery for.

If $\rho \geq \frac{1}{2}$ then the probability of producing a collision in time t_{cr} is $\rho\epsilon \geq \frac{\epsilon}{2} \geq 2\epsilon_{cr}/2 = \epsilon_{cr}$. Contradiction with the assumption that h is (t_{cr}, ϵ_{cr}) hash function.

If on the other hand $\rho < \frac{1}{2}$, we know that the probability of producing an ots-forgery in time t_{ow} is $(1 - \rho)\epsilon/2^N \geq \epsilon/2^{N+1}$.

We have shown that by assuming $Pr(\text{"mssforgery"}) \geq 2\max\{\epsilon_{cr}, 2^N \cdot \epsilon_{ots}\}$ within time $\max\{t_{cr} - (N+1) \cdot t_H, t_{ots}\} - t_G - 2^N \cdot t_S$ we get $Pr(\text{"collision"}) \geq \epsilon_{cr}$ within time t_{cr} or $Pr(\text{"otsforgery"}) \geq \epsilon_{ots}$ within time t_{ots} . \square

In order to estimate the security of the one-time signature scheme the following theorems from [7] will be very useful:

Theorem 1.23. [7] *If there exists a polynomial-time adversary that, after seeing a signature $\tilde{s} = \tilde{k} \cdot z + \tilde{l}$ of a message z , can output a valid signature of another message z' with probability $1/\text{poly}(n)$, then there exists a polynomial time algorithm that can solve the $Col_{d, H_{R,m}}(h)$ problem for $d = 10 \cdot \phi \cdot p^{1/m} \cdot n \cdot \log^2 n$.*

We will use the idea of the proof of the previous result to establish and prove an useful result that we will use later in section 2.

Theorem 1.24. *If there exists an adversary that, after seeing a signature $\tilde{s} = \tilde{k} \cdot z + \tilde{l}$ of a message z , can output a valid signature of another message z' , then with probability almost equal to 1, one can also solve the $Col_{d, H_{R,m}}(h)$ problem for $d = 10 \cdot \phi \cdot p^{1/m} \cdot n \cdot \log^2 n$.*

Remark. The following proof of the theorem is almost the same as the one given in [7]. There are only few places where changes were needed in order to adapt the proof to the slightly changed theorem.

Proof Δ (Mainly the proof from [7] with slight changes) Let A be an adversary who can break the one-time signature scheme. This means that after seeing a signature for any message of his choice, A can then successfully sign a different message of his choice. We'll be assuming (as in the original proof) that $\tilde{k} \in DK_{\lfloor \log^2(n)-1 \rfloor}$ and $\tilde{l} \in DL_{\lfloor \log^2(n)-1 \rfloor}$, where the sets DK and DL are defined as in 5.

The algorithm below uses the message-forging adversary A to solve the $Col_{d, H_{R,m}}(h)$ problem for the parameters specified in the previous theorem.

Algorithm for solving $Col_{d, H_{R,m}}(h)$

- Run the Key-Generation algorithm (but use the given h instead of generating a random one). We generate \hat{k} and \hat{l} .
- Receive message z from A .
- Send $\hat{k}z + \hat{l}$ to A .
- Receive message z' and its signature \hat{s}' from A .
- Output \hat{s}' and $\hat{k}z' + \hat{l}$.

The total time necessary for the running of the algorithm is the time to generate \widehat{k} and \widehat{l} (time bounded by $2 \cdot m \cdot n \cdot \text{Rand}(80 \cdot \log^2(n) \cdot n^{1+\frac{3}{\log(n)}})$ (where $\text{Rand}(n)$ is the time to generate a random integer with modulus $\leq n$ under the probability P (for the probability see section 5))) plus the time to calculate two times the signature of a message (bounded by $4 \cdot n^2$) plus the time for running of the A 's signature forging algorithm plus the time for which we will note $\text{Sigforge}(n)$.

So the total time necessary for the previous algorithm will be $(O(n^3) + \text{Sigforge}(n))$. And since the problem of forging this particular type of Micciancio-Lyubashevsky's signatures is considered to be a difficult non-polynomial we can well approximate the time complexity of the algorithm for solving $\text{Col}_{d,H_{R,m}}(h)$ by $\text{Sigforge}(n)$.

The rest of the proof is the same as for the initial theorem (see ??) but we will put it here in order to make the proof complete.

We now need to show that the outputs of the above algorithm are a collision for the function h with non-negligible probability. If A forges a signature \widehat{s}' for z' , then $\|\widehat{s}'\|_\infty \leq 10 \cdot p^{1/m} \cdot n \cdot \log^2 n$ and

$h(\widehat{s}') = h(\widehat{k})z' + h(\widehat{l}) = h(\widehat{k}z' + \widehat{l})$. So if $\widehat{s}' \neq \widehat{k}z' + \widehat{l}$, then our algorithm outputted two distinct elements that form a collision for the function h . On the other hand, if $\widehat{s}' = \widehat{k}z' + \widehat{l}$, then we do not get a collision. To complete the proof of the theorem, we will show that it's extremely unlikely that an adversary (even one with unlimited computational power) can produce \widehat{s}' and a z' such that $\widehat{s}' = \widehat{k}z' + \widehat{l}$. This will be done in two steps. In the first step, we show that being able to produce such an \widehat{s}' and a z' implies uniquely determining the signing key $(\widehat{k}, \widehat{l})$. Then in the second step we show that given the public key $(h, h(\widehat{k}), h(\widehat{l}))$ and a signature $\widehat{k}z + \widehat{l}$ of message z , it is information theoretically impossible to determine the signing key $(\widehat{k}, \widehat{l})$. This means that if A is able to forge a signature \widehat{s}' for some message z' , then almost certainly $\widehat{s}' \neq \widehat{k}z' + \widehat{l}$. We now show that obtaining an \widehat{s}' and a z' such that $\widehat{s}' = \widehat{k}z' + \widehat{l}$ uniquely determines \widehat{k}, \widehat{l} . Since we know that $\widehat{s} = \widehat{k}z + \widehat{l}$ and $\widehat{s}' = \widehat{k}z' + \widehat{l}$, it follows that $\widehat{s} - \widehat{s}' = \widehat{k}(z - z')$. Since $\|\widehat{k}\|_\infty \leq 5 \cdot p^{1/m} \log^2 n$ and $\|z - z'\|_\infty \leq 2$, multiplying \widehat{k} by $z - z'$ in the ring $\mathbb{Z}_p[x]/\langle f \rangle$ is the same as multiplying them in the ring $\mathbb{Z}_p[x]/\langle f \rangle$ because the coefficients never get big enough to get reduced modulo p . This is because $\|\widehat{k}(z - z')\|_\infty \leq 10 \cdot p^{1/m} \cdot n \cdot \log^2 n$, but in order to get reduced modulo p , the absolute value of the coefficients would have to be at least $p/2$, which is a much larger quantity. Now, since the ring $\mathbb{Z}[x]/\langle f \rangle$ is an integral domain and $z - z' \neq 0$, there cannot exist another key $\widehat{k}' \neq \widehat{k}$ such that $\widehat{k}'(z - z') = \widehat{k}(z - z')$. And so the key \widehat{k} is uniquely determined (and is equal to $\frac{\widehat{s} - \widehat{s}'}{z - z'}$), and similarly the key $\widehat{l} = \widehat{s} - \widehat{k}z$ is also unique.

Now we move on to showing that by knowing only $h, h(\widehat{k}), h(\widehat{l}), z$, and $\widehat{k}z + \widehat{l}$, it is information theoretically impossible to determine the signing key $(\widehat{k}, \widehat{l})$ (and thus, information theoretically impossible to come up with \widehat{s}', z'

such that $\widehat{s}' = \widehat{k}z' + \widehat{l}$. The idea is to show that for every $h, h(\widehat{k}), h(\widehat{l}), z, \widehat{k}z + \widehat{l}$ there is an exponential number of signing keys $(\widehat{k}', \widehat{l}')$, other than $(\widehat{k}, \widehat{l})$, that satisfy $h(\widehat{k}) = h(\widehat{k}')$, $h(\widehat{l}) = h(\widehat{l}')$, and $\widehat{k}z + \widehat{l} = \widehat{k}'z + \widehat{l}'$. And in addition, the total probability that one of these other keys was chosen in the key-generation step (conditioned on $h, h(\widehat{k}), h(\widehat{l}), z, \widehat{k}z + \widehat{l}$) is almost one.

We point out that we are not proving witness-indistinguishability. It's actually quite possible that for every other key $(\widehat{k}', \widehat{l}')$, the probability that it was the key that was used to sign the message is exponentially smaller than the probability that $(\widehat{k}, \widehat{l})$ was the key. What we will be showing is that the sum of probabilities of all other possible keys combined being the secret key is exponentially larger than the probability that $(\widehat{k}, \widehat{l})$ was the key.

Lemma 1.25. *(For the proof please see [7]) Let (h, K, L) be the verification key of the signature scheme and \widehat{s} is the signature of some message z . Then for any signing key $(\widehat{k}, \widehat{l})$ such that $\widehat{k} \in DK_{\lfloor \log^2 n - 1 \rfloor}$, $\widehat{l} \in DK_{\lfloor \log^2 n - 1 \rfloor}$, $h(\widehat{k}) = K$, $h(\widehat{l}) = L$, and $\widehat{s} = \widehat{k}z + \widehat{l}$, the probability that this was the actual signing key generated by the key-generation algorithm is negligibly small.*

□

An important corollary of this theorem is the following one:

Corollary 1.26. *Complexity for forging a MLots \approx Complexity for solving the $Col_{d, \mathcal{H}_{R,m}}(h)$ problem for $d = 10 \cdot \phi \cdot p^{1/m} \cdot n \cdot \log^2 n$.*

Theorem 1.27. *[7] Let f be an irreducible polynomial in $\mathbb{Z}[x]$ of degree n and define integers $p = (\phi n)^3$ and $m = \lceil \log n \rceil$. If there exists a polynomial-time algorithm that solves $Col_{d, \mathcal{H}_{R,m}}(h)$ for $R = \mathbb{Z}_p[x]/\langle f \rangle$ and $d = 10\phi p^{1/m} n \log^2 n$, then there exists a polynomial-time algorithm that approximates $\lambda_1(\mathcal{L})$ to within a factor of $\widetilde{O}(\phi^5 n^2)$ for every lattice \mathcal{L} corresponding to an ideal in the ring $\mathbb{Z}[x]/\langle f \rangle$.*

The previous two theorems when combined together prove that an attacker who can forge the one-time signature will be able to solve a SPP_γ for some small value of γ , which is supposed difficult (see [4]).

Very Important Remark: If we combine only the first of these two theorems with the main theorem about the MLMSS security (see 1.22) we obtain that the security of the MLMSS is based solely on the difficulty to find collisions for the Micciancio-Lyubashevsky's hash function.

This result will be of extreme importance in estimating the security of MLMSS because instead of estimating separately the security of Micciancio-Lyubashevsky's one time signature and the collision-resistance of the Micciancio-Lyubashevsky's hash function and combining this with the MLMSS security theorem (see 1.22), what we will need in reality will be only estimating the security (collision-resistance) of the Micciancio-Lyubashevsky's hash function!

For the estimation of the security of collision resistance of the Micciancio-Lyubashevsky's hash the following theorem from [4] and more particularly its proof will be useful:

Theorem 1.28. *Let \mathcal{H} be a hash function family as above with $m > \frac{\log_2(p)}{\log_2(2 \cdot d)}$ and $p > 2 \cdot \text{EF}(f, 3) \cdot d \cdot m \cdot n^{1.5} \log n$. Then, for $\gamma = 8 \cdot \text{EF}(f, 3)^2 \cdot d \cdot m \cdot n \cdot \log^2 n$, there is a polynomial time reduction from $f - \text{SPP}_\gamma(I)$ for any I to $\text{Col}_{\mathcal{H}}(h)$ where h is chosen uniformly at random from \mathcal{H} .*

We will enter in the details of the proof of this theorem in the section 3.

1.12 Some approximations based on the results from the previous section

On the basis of the result from Kumar and Sivakumar 1.17 (see previous section) we could make a useful conjecture about the security of the Micciancio-Lyubashevsky-Merkle signature scheme. The conjecture we will be that no one can find a generic solution to SVP_γ with $\gamma = n^4$ in time less than $2^{O(n)}$ (we took $\epsilon = 1$ in the above proposition and we obtained thus a $2^{O(n)}$ time algorithm for approximating the shortest vector in an n -dimensional lattice within a factor of n^4).

As the approximation factor that we need for the Micciancio-Lyubashevsky hash function is $\gamma = 8 \cdot \text{EF}(f, 3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2 = \tilde{O}(n) \cdot \text{EF}(f, 3)^2 \lesssim n^4$ we could reasonably suppose that we will need at least $2^{O(n)}$ time to find an approximate of the shortest vector in the corresponding lattice problem. The same conclusion is true as well for the Micciancio-Lyubashevsky's one-time signature where the approximation factor $\gamma = 80 \cdot \phi^3 \cdot n^2 \cdot (\phi n)^{\frac{3}{\lceil \log n \rceil}} \cdot \log_2^5(n) \lesssim n^4$. So finding an approximate shortest vector in the lattice problems for both Micciancio-Lyubashevsky's hash function and Micciancio-Lyubashevsky's one-time signature will take time at least $2^{O(n)}$.

Proposition 1.29. *There is no algorithm which runs in time less than $2^{O(n)}$ and which could find a collision for the Micciancio-Lyubashevsky hash function or forge the Micciancio-Lyubashevsky one time signature.*

Due to the fact that this result does not provide information about the hidden factor in the big O notation, we will choose a value for the practical evaluations (e.g. 1). This is certainly an ambitious assumption but which we will not use in establishing the general form of the security of the MLMss (see) and which we will use only in some of the practical results about the security of the scheme.

2 The main result of this thesis

The main result of this thesis is the following one which quantifies the security of MLMSS as a function of the difficulty to find, given an ideal

$I \subseteq \mathbb{Z}^n / \langle f \rangle$ where f is a monic polynomial of degree n , a polynomial $g \in I$ such that $g \neq 0$ and $\|g\|_f \leq n^4 \lambda_1^\infty(I)$ (here $\lambda_1^\infty(I)$ denotes the shortest polynomial for the infinity norm in the lattice $\mathcal{L}(I)$). This problem is called the shortest polynomial problem SPP_γ with approximation factor γ (above $\gamma = n^4$) and is supposedly as difficult as the $SV P_{n^4}$ (in general $SV P_\gamma$) (see [4]).

More precisely I quantify the security of the MLMSS as a function of the complexity of an $f - SPP_\gamma$ problem. Please come to the thesis presentation for more details on this choice. I hope I will be able to put a draft of the presentation also online.

So in plain words the following important theorem states that if the problem of finding of the shortest polynomial within approximation factor of n^4 in lattices $\mathcal{L}(I)$ (see above) is a difficult one, then the problem of forging the MLMss will also be a difficult problem:

Theorem 2.1 (Security of MLMss). *Let n_{crh}, n_{ots}, N be the parameters of an MLMss. Then for some irreducible polynomial f of degree $n = \max\{n_{crh}, n_{ots}\}$, we may consider the corresponding $f - SPP_{n^4}$ -problem and set*

$$t = \max\left\{ \frac{1}{1.465} \cdot \left(\mathcal{C}(f - SPP_{n_{crh}^4}) / \left(\frac{y(n_{crh})^2}{y(n_{crh})-1} \right) \right)^{1/2} - (N+1) \cdot t_H, \frac{1}{1.465} \cdot \left(\mathcal{C}(f - SPP_{n_{ots}^4}) / \left(\frac{y(n_{ots})^2}{y(n_{ots})-1} \right) \right)^{1/2} \right\} - t_G - 2^N \cdot t_S$$

$$\epsilon = 2 \cdot \max\left\{ \left(\mathcal{C}(f - SPP_{n_{crh}^4}) / \left(2^5 \cdot \frac{y(n_{crh})^2}{y(n_{crh})-1} \right) \right)^{-1/2}, 2^N \cdot \left(\mathcal{C}(f - SPP_{n_{ots}^4}) / \left(2^5 \cdot \frac{y(n_{ots})^2}{y(n_{ots})-1} \right) \right)^{-1/2} \right\}$$

and if $t > 0$ and $\epsilon \leq 1$, then MLMss with these parameters is $(t, \epsilon, 2^N)$ existentially unforgeable under adaptive chosen message attacks.

Here $y(n) = (\lceil n/5 \rceil - \log_2(n^4))$ (we have the following inequalities for $y(n)$: $2 \leq y(n) \leq n/5$)

Moreover t_G is the time taken to generate an MLMss instance, t_S is the time taken for signing a message using MLMss, and t_H is the time taken for hashing.

Proof Δ The proof of the MLMss security will take several steps which we describe below:

1. First we will use the result from Theorem 1.22 to base the security of MLMss Security on the difficulty to find a collision for the hash functions presented in the section 1.8.
2. Second we will prove that finding a collision for the hash functions has a comparable complexity to the $f - IncSPP_{n^4}$ - problem (described previously: see 1.15).
3. Third we will use the reduction from (Lemma 3.2): $f - IncSPP_{n^4} - > SPP_{n^4}$

So lets start with the proof and let $H(R, D, m)$ (for its definition see section 1.8) be a (t_{cr}, ϵ_{cr}) (see definition 1.4) collision resistant family of Micciancio-Lyubashevsky's hash-functions (as introduced in section 1.8 and used in [4]), and let MLots be a Micciancio-Lyubashevsky one-time signature $(t_{ots}, \epsilon_{ots})$ one-time signature (as introduced in 1.9 and used in [7]).

Then for any number N (i.e. for any height of the Micciancio-Lyubashevsky-Merkle tree) with a direct application of theorem 1.22 which was already proved in section 1, we obtain that the Micciancio-Lyubashevsky-Merkle signature is $(t, \epsilon, 2^N)$ existentially unforgeable under adaptive chosen message attacks, where t and ϵ given respectively by the expressions:

$$t := \max\{t_{cr} - (N + 1) \cdot t_H, t_{ots}\} - t_G - 2^N \cdot t_S$$

$$\epsilon := 2\max\{\epsilon_{cr}, 2^N \cdot \epsilon_{ots}\}$$

should satisfy the inequalities $t > 0$ and $\epsilon \leq 1$. For the definition of the terms t_G , t_S and t_H please read the statement of this theorem (they are explained there).

Now in the next step we will use the theorem 1.23 ([7]) to establish a relationship between the time complexity of forging a MLots and the time complexity of finding a collision for the MLots hash function. The theorem 1.24 and its corollary 1.26 state that the time complexity for finding an efficient algorithm for forging the Micciancio-Lyubashevsky signature is approximately the same as the time complexity for finding a collision for the hash function presented in section 1.8 ([7]).

So the following result should be satisfied:

$$\mathcal{C}(\text{MLots}) \approx \mathcal{C}(\text{MLots hash function}) \quad (1)$$

Now we have an implicit relationship between the security of MLMss and the security of both hash functions presented in section 1.8.

We will now establish a relationship between the security of the hash functions from section 1.8 and the difficulty of a problem called $f - \text{IncSPP}_\gamma$ which was already presented in 1.15. To accomplish this goal we will use the corollary 3.1 which will be proved in section 3. The result is the following:

$$(\text{Expression of the time complexity of } f - \text{IncSPP}_{n^4} \text{ (see 3.1)})$$

The time complexity of $f - \text{IncSPP}_\gamma$ is given by:

$$\log_2(n) \cdot \mathcal{O}(n^3) + \mathcal{C}(\text{Col}(n))$$

,where n is the security parameter (it can as well be n_{crh} or n_{ots} (see 1.10) and $\mathcal{C}(\text{Col}(n))$ is the time necessary to find a collision for the corresponding Micciancio-Lyubashevsky's hash function (see section 1.8). (For more details on the result please read 3.1)

If we take into account in this expression also that the problem of finding collisions should be a difficult one (and certainly not a polynomial one with exponent equal only to 3) we obtain a useful approximation of the previous result:

$$\mathcal{C}(f - IncSPP_\gamma) \approx \mathcal{C}(Col(n))$$

In this expression it will be interesting to write the difficulty for finding a collision as a function of the time complexity of the $f - IncSPP_\gamma$ -problem:

$$\mathcal{C}(Col(n)) \approx \mathcal{C}(f - IncSPP_\gamma)$$

So taking into account the previous approximation we obtain:

$$\mathcal{C}(\text{ML hash function}), \mathcal{C}(\text{MLots hash function}) \approx \mathcal{C}(f - IncSPP_\gamma) \quad (2)$$

In the next step we will use the following result (please see the proof in 3.2) which gives us a relationship between the complexities of $f - SPP_\gamma$ and $f - IncSPP_\gamma$ problems:

Polynomial time reduction from $f - SPP_\gamma$ to $f - IncSPP_\gamma$

Lemma 2.2. *(For the proof please see 3.2) There is a polynomial time reduction from $f - SPP_\gamma$ to $f - IncSPP_\gamma$ and for the complexity of $f - SPP_{\gamma(n)}$ we have:*

$$\mathcal{C}(f - SPP_{\gamma(n)}) \approx n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n - 1) - \log_2(n^4) \rceil \cdot \mathcal{C}(f - IncSPP_{\gamma(n)})$$

Remark. Please note that the approximation $\log_2(\gamma(n)) \approx \log_2(n^4)$ that we used for the expressions of $\gamma(n_{crh})$ and $\gamma(n_{ots})$ is too simplified for practical use. We use it here only in order to make the main result more readable.

Using the previous inequality and writing the complexity of $f - IncSPP_{\gamma(n)}$ as a function of complexity of $f - SPP_{\gamma(n)}$ we obtain that if the inequality $\lceil 0.2075185 \cdot (n - 1) - \log_2(n^4) \rceil > 0$ is satisfied then:

$$\mathcal{C}(f - IncSPP_{n^4}) \approx (\mathcal{C}(f - SPP_{n^4}) - n^3 \cdot \log(n)) / (\lceil 0.2075185 \cdot (n - 1) - \log_2(n^4) \rceil)$$

We can simplify again a bit the expression taking into account that the $(f - SPP_{n^4})$ -problem is a difficult problem (and certainly not a polynomial one with exponent equal to 3). So as a result we obtain the following simplified expression:

$$\mathcal{C}(f - IncSPP_{n^4}) \approx \mathcal{C}(f - SPP_{n^4}) / (\lceil 0.2075185 \cdot (n - 1) - \log_2(n^4) \rceil) \quad (3)$$

So we will now use some of the previous inequalities and more precisely inequalities (1), (2) and (3) and combine them to obtain the following slightly simplified inequalities:

$$\mathcal{C}(\text{MLots}) \approx \mathcal{C}(f - SPP_\gamma) / (\lceil (n_{ots}/5) - \log_2(n_{ots}^4) \rceil) \quad (4)$$

and

$$\mathcal{C}(\text{ML hash function}) \approx \mathcal{C}(f - SPP_\gamma) / (\lceil (n_{crh}/5) - \log_2(n_{crh}^4) \rceil) \quad (5)$$

Taking into account that the denominator is the same in both of the expressions we can introduce $y(n) = (\lceil (n/5) - \log_2(n^4) \rceil)$ to simplify the previous expressions:

$$\mathcal{C}(\text{MLots}), \mathcal{C}(\text{ML hash function}) \approx \mathcal{C}(f - SPP_\gamma) / y(n) \quad (6)$$

where the inequality is true both for n equal n_{crh} and n equal n_{ots} .

Taking into account that the inequality $(\lceil (n/5) - \log_2(n^4) \rceil) \geq 2$ is satisfied, we can determine the following bounds for $y(n)$:

$$2 \leq y(n) \leq \frac{n}{5}$$

If instead of finding $(\lceil (n/5) - \log_2(n^4) \rceil)$ collisions of the hash function, we find only $(\lceil (n/5) - \log_2(n^4) \rceil) - 1$ collisions (i.e. if we equivalently solve only $(\lceil (n/5) - \log_2(n^4) \rceil) - 1$ times the $f - IncSPP_\gamma$ -problem) we will find certainly a solution to the $SPP_{2\gamma}$ -problem but not necessary to the SPP_γ -problem. The probability that by solving the $SPP_{2\gamma}$ -problem, we have also solved the SPP_γ -problem (if we suppose that the coordinates of the vector solution to the $SPP_{2\gamma}$ are uniformly distributed in the interval $[0, 2 \cdot \gamma]$ and taking into account that we have n coordinates) is approximately $\frac{1}{2^n}$

So using the notation (t, ϵ) , we have obtain that:

The problem $f - SPP_\gamma$ is $(\mathcal{C}(f - SPP_{2\gamma}), \frac{1}{2^n})$ -secure.

Now using the fact that for solving the $f - SPP_{2\gamma}$ we have to find $y(n) - 1$ collisions, while for solving the $f - SPP_\gamma$ we have to find $y(n)$ collisions we have the following relations between the complexities of $f - SPP_{2\gamma}$ and $f - SPP_\gamma$:

$$\mathcal{C}(f - SPP_{2\gamma}) / (y(n) - 1) = \mathcal{C}(f - SPP_\gamma) / y(n)$$

So writing the the previous result in terms of the complexity of the $f - SPP_\gamma$ -problem we obtain that:

The problem $f - SPP_\gamma$ is $(\frac{(y(n)-1)}{y(n)} \cdot \mathcal{C}(f - SPP_\gamma), \frac{1}{2^n})$ -secure.

Now taking into account that for solving $f - SPP_\gamma$ are necessary to find $y(n)$ collisions for the hash function and supposing that each of the collisions is independant of the others we obtain that:

Finding a collision is $(\frac{y(n)-1}{y(n)^2} \cdot \mathcal{C}(f - SPP_\gamma), \frac{1}{2^{\frac{n}{y(n)}}})$ -secure.

The last statement can be reformulated taking into account that the complexity of forging MLots, and the problem of forging the ML hash function are equivalent in complexity to finding a collision for the corresponding hash function. Moreover we have that $y(n) < \frac{n}{5}$ and therefore $\frac{1}{2^{\frac{n}{y(n)}}} < \frac{1}{2^5}$. So taking these remarks into consideration we obtain:

the problem of forging MLots, and the problem of forging the ML hash function are $(\frac{y(n)-1}{y(n)^2} \cdot \mathcal{C}(f - SPP_\gamma), \frac{1}{2^5})$ -secure.

The last result that we will use is a small lemma which enables us to convert a problem which is (t, ϵ) -secure into a (t', ϵ') -secure one. For the proof of the result please see theorem 6.2. The remark after 6.2 enables us to convert in particular a (t, ϵ) -secure problem into a $((\frac{t}{\epsilon})^{1/2}, (\frac{t}{\epsilon})^{-1/2})$ -secure one for large values of the ratio $(\frac{t}{\epsilon})$ and this will be particularly interesting here. Using this result and the previous result (see above) we obtain that the problems $(t_{crh}, \epsilon_{crh})$ and $(t_{ots}, \epsilon_{ots})$ can be written respectively as:

$$(t_{crh}, \epsilon_{crh}) = (\frac{1}{1.465} \cdot ((\mathcal{C}(f - SPP_\gamma) / (\frac{y(n_{crh})^2}{y(n_{crh})-1}))^{1/2}, ((\mathcal{C}(f - SPP_\gamma) / (2^5 \cdot \frac{y(n_{crh})^2}{y(n_{crh})-1}))^{-1/2})$$

and

$$(t_{ots}, \epsilon_{ots}) = (\frac{1}{1.465} \cdot ((\mathcal{C}(f - SPP_\gamma) / (\frac{y(n_{ots})^2}{y(n_{ots})-1}))^{1/2}, ((\mathcal{C}(f - SPP_\gamma) / (2^5 \cdot \frac{y(n_{ots})^2}{y(n_{ots})-1}))^{-1/2})$$

Remark: For the term the origin of the term $\frac{1}{1.465}$ please see the next section on the approximations.

Lastly we use the previous expressions to obtain the main result:

$$t = \max\{\frac{1}{1.465} \cdot ((\mathcal{C}(f - SPP_\gamma) / (\frac{y(n_{crh})^2}{y(n_{crh})-1}))^{1/2} - (N + 1) \cdot t_H, \frac{1}{1.465} \cdot ((\mathcal{C}(f - SPP_\gamma) / (\frac{y(n_{ots})^2}{y(n_{ots})-1}))^{1/2}\} - t_G - 2^N \cdot t_S$$

$$\epsilon = 2 \cdot \max\{((\mathcal{C}(f - SPP_\gamma) / (2^5 \cdot \frac{y(n_{crh})^2}{y(n_{crh})-1}))^{-1/2}, 2^N \cdot ((\mathcal{C}(f - SPP_\gamma) / (2^5 \cdot \frac{y(n_{ots})^2}{y(n_{ots})-1}))^{-1/2}\}$$

□

2.1 The approximations used in the proof of the main theorem

We have used the following approximations:

1. An important assumption:

We supposed that someone who has solved the $f - SPP_{2\gamma}$ -problem has a probability equal to $\frac{1}{2^n}$ to have solved by the solution to $f - SPP_{2\gamma}$ -problem as well the $f - SPP_\gamma$ -problem.

2. In the expression:

$$\mathcal{C}(f - IncSPP_\gamma) = \log_2(n) \cdot \mathcal{O}(n^3) + Col(n) \quad (*)$$

where the hidden factor in the $\mathcal{O}(n^3)$ is inferior to 1 (see 3) we have made the approximation:

$$\mathcal{C}(f - IncSPP_\gamma) \approx \mathcal{C}(Col(n))$$

Doing this we made an error $\frac{\log_2(n) \cdot \mathcal{O}(n^3)}{Col(n)}$ on the time complexity of $f - IncSPP_\gamma$.

We take into account that $y(n) \cdot (\log_2(n) \cdot \mathcal{O}(n^3) + \mathcal{C}(Col(n))) = \mathcal{C}(f - SPP_\gamma)$ and that $y(n) \leq \frac{n}{5}$. We obtain that $y(n) \cdot \log_2(n) \cdot \mathcal{O}(n^3) \leq \log_2(n) \cdot \frac{n^4}{5}$. For typical values for the parameter n we have $n \approx 4000$ and in this case $\log_2(n) \cdot \frac{n^4}{5} \approx 2^{49}$ and therefore the error caused by the above approximation is bounded by $2^{49}/2^{80} \approx 2^{-31}$ taking into account that the problem $f - SPP_\gamma$ should have time complexity at least 2^{80} .

So by making the approximation (*) for $n \approx 4000$ we made an error bounded by 2^{-30} .

3. In the expression:

$$\mathcal{C}(f - SPP_{\gamma(n)}) = n^3 \cdot \log(n) + [0.2075185 \cdot (n - 1) - \log_2(n^4)] \cdot \mathcal{C}(f - IncSPP_{\gamma(n)})$$

we neglected the term $n^3 \cdot \log(n)$ and we wrote:

$$\mathcal{C}(f - SPP_{\gamma(n)}) = [0.2075185 \cdot (n - 1) - \log_2(n^4)] \cdot \mathcal{C}(f - IncSPP_{\gamma(n)}) \quad (**)$$

Making this approximation we made an error equal to $\frac{n^3 \cdot \log(n)}{f - SPP_{\gamma(n)}}$. For $n \approx 4000$ and taking into account that the time complexity of the $f - SPP_\gamma$ -problem is at least 2^{80} we obtain that the error made is bounded by $2^{39}/2^{80} \approx 2^{-41}$.

So by making the approximation (**) for $n \approx 4000$ we made an error bounded by 2^{-40} .

4. In the expression:

$$\mathcal{C}(f - IncSPP_{n^4}) \approx \mathcal{C}(f - SPP_{n^4}) / ([0.2075185 \cdot (n - 1) - \log_2(n^4)])$$

we approximated $0.2075185 \cdot (n - 1)$ by $\frac{n}{5}$ (***) .

Doing this we made an error equal to $\frac{0.2075185 \cdot (n-1) - 0.2 \cdot n}{0.2 \cdot n}$ on the value of the denominator. So for $n = 4000$ this error is 0.035 which is inferior to 0.05 . If a precision of 0.035 is of importance we should use the expression $0.2075185 \cdot (n - 1)$ instead of $\frac{n}{5}$. In the main theorem we did this approximation only in order to make the main theorem more readable.

So by making the approximation (***) for $n \approx 4000$ we made an error bounded by 0.035 .

5. We used an approximation in the last step when we converted a problem which was $(t, \frac{1}{2^5})$ -secure into a $(t^{\frac{1}{2}}, t^{-\frac{1}{2}} \cdot \frac{1}{2^5})$ -secure We use for the conversion the following theorem that we prove later:

Theorem 2.3. *If a problem is $(t, 1 - \epsilon)$ secure for some $\epsilon > 0$, then for any $1 > \delta > 0$ we can set $k = \lceil \frac{\log(\epsilon)}{\log(1-\delta)} \rceil$ and the same problem is $(t/k, \delta)$ secure.*

To apply the theorem we choose $\epsilon = \frac{31}{32}$ and $\delta = t^{-\frac{1}{2}} \cdot \frac{1}{2^5}$. We obtain that $k \approx \lceil -0.0458 / -t^{-\frac{1}{2}} \cdot \frac{1}{2^5} \rceil = \lceil 1.465 \cdot t^{\frac{1}{2}} \rceil \approx 1.465 \cdot t^{\frac{1}{2}}$.

In the previous expressions we have used the Taylor developpement:

$$\log(1 - x) = - \sum_{n=1}^{\infty} \frac{x^n}{n} \text{ for } |x| < 1$$

and we made the approximation:

$$\log(1 - x) \approx -x \text{ (****)}$$

so that we neglected all the terms in x^2 and more. To estimate what error have caused this approximation we can simply calculate the error caused by the predominant term of the rest which is $\frac{x^2}{2 \cdot x} = \frac{x}{2}$. Taking into account that $t = \mathcal{C}(f - SPP_\gamma) / (\frac{y(n_{ots})^2}{y(n_{ots})-1})$, $x = t^{-1/2}$ and $\mathcal{C}(f - SPP_\gamma) \geq 2^{80}$ we obtain again for $n = 4000$ that $x \leq 2^{-35}$ and therefore the error caused by the approximation is inferior to 2^{-36} .

So by making the approximation (****) for $n \approx 4000$ we made an error bounded by 2^{-36} .

2.2 What we will see in the next sections

In the following sections we will establish some of the results that we used in this section. We will also a sometimes useful estimation for the difficulty of obtaining collisions, we will then discuss consecutively the security of the Micciancio-Lyubashevsky's hash function and the Micciancio-Lyubashevsky's one time signature against LLL-based 1.18, Schnorr-BKZ-based 1.19, Koy's primal-dual-based 1.20 and primal-dual RSR-based 1.21 attacks. We will also see that each one of the previous attacks will demand bigger values for the security parameters than the previous ones until we end up with the attacks based on the primal-dual RSR algorithm the security of

which will determine the minimal requirements for the security parameters (please see 6.1). We will also see that the attacks based on the LLL algorithm (or using it in some of their steps) are in practice more dangerous than their worst case theoretical formulation may suggest (see 4.2, 5.2). Finally in the last section we will provide some theoretical and practical results about the security of MLMSS against various attacks.

3 A useful inequality estimation for the difficulty of finding a collision for the Micciancio-Lyubashevsky's hash

The algorithm for solving $f - IncSPP_\gamma$ with probability ≥ 0.01 proposed in [4] (in the work of Peikert and Rosen "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices" Lemma 2.10 is proven that this probability is almost equal to 1) consists of the following steps:

(1) For $i = 1$ to m

(2) Generate a uniformly random coset of $I/\langle g \rangle$ and let v_i be a polynomial in that coset. (This step have a time complexity of $\mathcal{O}(n^2)$ (please see the Appendix Lemma A.1.))

(3) Generate $y_i \in \mathbb{R}^n$ such that y_i has distribution ρ_s/s^n and consider y_i as a polynomial in $\mathbb{R}[x]$. (This step have a time complexity of $\mathcal{O}(n)$ (please see the Appendix Lemma A.2.))

(4) Let w_i be the unique polynomial in $\mathbb{R}[x]$ of degree less than n with coefficients in the range $[0, p)$ such that $p \cdot (v_i + y_i) \equiv g \cdot w_i$ in $\mathbb{R}^n/\langle pg \rangle$. (This step have a time complexity of $\frac{2n^3}{3} + \mathcal{O}(n^2 \cdot \log(n) \cdot \log\log(n))$ (please see the Appendix Lemma A.3.))

(5) $a_i = [w_i] \bmod p$ (where $[w_i]$ means round each coefficient of w_i to the nearest integer). (This step have a time complexity of $\mathcal{O}(n)$ (please see the Appendix Lemma A.4.))

(6) call oracle $\mathcal{C}(a_1, \dots, a_m)$, and using its output, find polynomials z_1, \dots, z_m such that $\|z_i\|_f = 2d$ and $\sum z_i a_i \equiv 0$ in the ring $\mathbb{Z}_p[x]/\langle f \rangle$. (This step has an unknown time complexity due to the fact that we do not know the time for finding a collision for the hash function. The goal of this section is to provide an estimation of the above complexity)

(7) output $h = (\sum (\frac{g(w_i - [w_i])}{p} - y_i) z_i) \bmod f$. (For this output we have that first of all $h \in I$ (Lemma 5.4. [4]) and that with a probability negligibly different from 1 $\|h\|_f \leq \frac{\|g\|_\infty}{2}$ (see Peikert and Rosen "Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices" Lemma 2.10)).

The total time complexity of the $f - IncSPP_\gamma$ will be therefore:

$$m \cdot (\mathcal{O}(n^2) + \mathcal{O}(n) + \frac{2n^3}{3} + \mathcal{O}(n^2 \cdot \log(n) \cdot \log\log(n)) + \mathcal{O}(n)) + \mathcal{C}(Col(n))$$

,where $\mathcal{C}(\text{Col}(n))$ is the time necessary to find a collision for the hash function.

So we obtain directly the following corollary (where we take into account that $m = \log_2(n)$ (see [4],[7])) :

Corollary 3.1. (*Expression of the time complexity of $f - \text{IncSPP}_\gamma$*) The time complexity of $f - \text{IncSPP}_\gamma$ is given by:

$$\log_2(n) \cdot \mathcal{O}(n^3) + \mathcal{C}(\text{Col}(n))$$

,where n is the security parameter (it can be as well n_{crh} as n_{ots} (see 1.10) and $\mathcal{C}(\text{Col}(n))$ is the time necessary to find a collision for the corresponding Micciancio-Lyubashevsky's hash function (see section 1.8).

Remark: In the above expression for the time complexity there are some approximations using the big \mathcal{O} notation. We will see later that all the terms in $\tilde{\mathcal{O}}(n^3)$ (or less) can be neglected for the calculus of the security of the MLMSS and so the predominant term in the above expression will be $\mathcal{C}(\text{Col}(n))$.

The following lemma from [4] will be quite useful in this section. In [4] it was provided without proof. Here we give a proof because it provides us with information about the number of reduction steps necessary for solving the $f - \text{SPP}_\gamma$ problem, given that we could solve the $f - \text{IncSPP}_\gamma$ problem.

Lemma 3.2. *There is a polynomial time reduction from $f - \text{SPP}_\gamma$ to $f - \text{IncSPP}_\gamma$*

and for the complexity of $f - \text{SPP}_{\gamma(n)}$ we have:

$$\mathcal{C}(f - \text{SPP}_{\gamma(n)}) \approx n^3 \cdot \log(n) + [0.2075185 \cdot (n - 1) - \log_2(n^4)] \cdot \mathcal{C}(f - \text{IncSPP}_{\gamma(n)})$$

Proof Δ First we can suppose without limitations that $\gamma \geq 1$ (in fact in our problem $\gamma_{\text{crh}} = 8 \cdot \text{EF}(f, 3)^2 \cdot d \cdot m \cdot n_{\text{crh}} \cdot \log^2 n_{\text{crh}} \geq 1$, and $\gamma_{\text{ots}} = 80\phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{\lceil \frac{3}{\log n_{\text{ots}}} \rceil} \cdot \log_2^5(n_{\text{ots}}) \geq 1$).

First of all we use the Strong-Segment LLL algorithm from Koy and Schnorr 1.18 to find in time $O(n^3 \cdot \log(n))$ a $(\frac{4}{3})^{\frac{n-1}{2}}$ approximation of the $f - \text{SPP}_\gamma$ problem i.e. to find a vector $g \in I$ (for I an ideal of $Z[x]/\langle f \rangle$) so that $g \neq 0$ and

$$\|g\|_f \leq \left(\frac{4}{3}\right)^{\frac{n-1}{2}} \lambda_1(I) = 2^{0.2075185 \cdot (n-1)} \lambda_1(I).$$

Lets now suppose that we know an algorithm \mathcal{A} which can solve $f - \text{IncSPP}_\gamma$. We start by applying this algorithm on g to find $g_1 \in I$ so that

$$\|g_1\|_f \leq \frac{\|g\|_f}{2}$$

then we apply the \mathcal{A} on g_1 to find $g_2 \in I$ so that $\|g_2\|_f \leq \frac{\|g_1\|_f}{2}$ and so on. After at most $\lceil 0.2075185 \cdot (n-1) - \log_2(8\text{EF}(f, 3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2) \rceil$ calls on our algorithm \mathcal{A} we would have found g' which will be a solution for $f - SPP_\gamma$ problem:

$$\|g'\|_f \leq \gamma \lambda_1(I)$$

To sum up the time needed to find a solution for $f - SPP_\gamma$ with $\gamma_{\text{crh}} = 8\text{EF}(f, 3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2$ is:

$$\begin{aligned} \mathcal{C}(f - SPP_{\gamma(n)}) &= n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n-1) - \log_2(8\text{EF}(f, 3)^2 \cdot d \cdot \\ & m \cdot n \cdot \log(n)^2) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log\log(n) + n) + \mathcal{C}(\text{Col}(n))) \\ & \text{(see above for the origin of the term } m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \\ & \log\log(n) + n) + \mathcal{C}(\text{Col}(n))) \end{aligned}$$

If we simplify the previous expression taking into account the values for the parameters suggested in [4], [7] and if we make the additional assumption that $\gamma_{\text{crh}}, \gamma_{\text{ots}} \approx n^4$ we obtain the following simplified expression for the complexity of the $f - SPP_{\gamma(n)}$ -problem valable for both n_{crh} and n_{ots} that we will only use in the main theorem 2 but not in the practical calculus due to its simplified form:

$$n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n-1) - \log_2(n^4) \rceil \cdot (\log_2(n) \cdot O(n^3) + \mathcal{C}(\text{Col}(n)))$$

We can write the previous result also in the form:

$$\begin{aligned} \mathcal{C}(f - SPP_{\gamma(n)}) &\approx n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n-1) - \log_2(n^4) \rceil \cdot \\ & \mathcal{C}(f - \text{IncSPP}_{\gamma(n)}) \end{aligned}$$

Again we will use this result only in the main result 2 but not for practical calculus due to its simplified form.

□

Remark. The term 0.2075185 is an approximation of the term $\frac{1}{2} \cdot \log_2(\frac{4}{3})$ in the LLL-reduction 1.18 (we have $\frac{1}{2} \cdot \log_2(\frac{4}{3}) \approx 0.2075185$)

Remark. $\mathcal{C}(f - SPP_{\gamma(n)}) = n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n-1) - \log_2(8\text{EF}(f, 3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log\log(n) + n) + \mathcal{C}(\text{Col}(n)))$

is the time complexity for solving $f - SPP_\gamma$ for $\gamma = 8\text{EF}(f, 3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2$ (i.e. for the value of γ used for hashing the Merkle tree) when we start the reduction with the application of the Schnorr proposition for LLL algorithm.

There are two more important remarks here:

Remark. 1. The complexity for solving the $f - SPP_\gamma$ -problem relative to the Micciancio-Lyubashevsky one-time signature will be (again using the Schnorr proposition for LLL algorithm 1.6):

$$\mathcal{C}(f - SPP_{\gamma(n)}) = n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n - 1) - \log_2(80\phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{\lceil \frac{3}{\log n_{\text{ots}}} \rceil} \cdot \log_2^5(n_{\text{ots}})) \rceil \cdot (m \cdot (2n + n^2 \cdot (1 + \log(n) \cdot \log \log(n)) + \frac{2n^3}{3}) + \mathcal{C}(\text{Col}(n))).$$

Remark. 2. We can use a BKZ-type algorithm instead of the Schnorr LLL-algorithm 1.18 for the initial reduction. We have considered in the Preliminaries three BKZ-algorithms (the most efficient ones) see 1.19, 1.20 and an algorithm using a combination of these two see 1.21.

If we use Koy's primale-dual algorithm (see 1.20) we will obtain the following time complexity of the $f - SPP_{\gamma}$ -problem:

$$\mathcal{C}(f - SPP_{\gamma(n)}) = \mathcal{O}(n^3 k^{\frac{k}{2} + o(k)} + n^4) + \lceil \frac{n}{k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log \log(n) + n) + \mathcal{C}(\text{Col}(n)))$$

If we use Schnorr's RSR algorithm (see 1.19) we will obtain the following time complexity of the $f - SPP_{\gamma}$ -problem:

$$\mathcal{C}(f - SPP_{\gamma(n)}) = \mathcal{O}(n^3 (\frac{k}{6})^{\frac{k}{4}} + n^4) + \lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log \log(n) + n) + \mathcal{C}(\text{Col}(n)))$$

If we use the primal-dual RSR 1.21 we will obtain the following time complexity of the $f - SPP_{\gamma}$ -problem:

$$\mathcal{C}(f - SPP_{\gamma(n)}) = \mathcal{O}(n^2 \cdot \log_2(n) \cdot (\frac{k}{11})^{\frac{k}{4}}) + \lceil \frac{n}{2} \cdot \log_2(1.025) - \log_2(\gamma) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log \log(n) + n) + \mathcal{C}(\text{Col}(n)))$$

For the rest we will consider that the initial reduction is done using the 1.18 algorithm, but no matter what algorithm we use the following proposition should be satisfied:

Proposition 3.3. *If after the initial application of the lattice reduction algorithm, the approximation factor that we obtain is superior to the approximation factor γ of the problem $f - SPP_{\gamma}$, we should not be able to find a solution to $f - SPP_{\gamma}$ with $\gamma \lesssim n^4$ for sufficient large n in time less than $\mathcal{O}(2^n)$ (Kumar and Sivakumar (see Preliminaries)) 1.17 so from the inequality which is always true:*

$$T(n) \geq \mathcal{C}(f - SPP_{\gamma}) \text{ with } \gamma = \mathcal{O}(n^4)$$

where $T(n)$ is the actual difficulty of the underlying problems in the Micciancio-Lyubashevsky's one-time signature and hash function, follows that:

$$T(n) \geq 2^n$$

taking into account 1.17 with the expression of $T(n)$ dependent on the algorithm that we use for the initial reduction (see above for some possibilities for the algorithm).

This proposition will be especially useful for estimating the security against attacks using only LLL-reductions (such as 1.18). See the corresponding paragraphs: 4.1, 5.1.

We will see later in paragraphs 4.3, 5.3, 4.4, 5.4 that when considering the security against RSR-reductions the important and limiting element is no more the inequality $T(n) \geq 2^n$ but the value of the parameter k which is limited by the inequality:

$$k \cdot (3 + \log_2(k)) \leq n \text{ (see [12])}$$

See the corresponding paragraphs for more details.

4 Security of Micciancio-Lyubashevsky's hash function

4.1 Security of the Micciancio-Lyubashevsky's hash function when the attacker use a LLL type reduction algorithm 1.18

There is a lot of flexibility in the signature scheme because we can choose the parameters for the hash function used for hashing the leaves of the Merkle Signature Tree almost independently from the parameters used for the one-time signature scheme at the end nodes. In particular that is the main reason why we use the index ots to denote the security parameter of the MLots and why use different index crh for the security parameter in this section.

For the estimation of the security of the collision-resistance of the hash function we use the result from the previous section:

$$T(n_{crh}) \geq \mathcal{C}(f - SPP_\gamma) \text{ with } \gamma = \mathcal{O}(n^4)$$

which we rewrite using the hypothesis from 1.17 in the form:

$$T(n_{crh}) \geq 2^{n_{crh}}$$

We could write therefore the inequality:

$$n^3 \cdot \log(n) + \lceil 0.2075185 \cdot (n - 1) - \log_2(8 \cdot \text{EF}(f, 3))^2 \cdot d \cdot m \cdot n \cdot \log(n)^2 \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log\log(n) + n) + \mathcal{C}(\text{Col}(n))) \geq 2^n$$

We should choose n sufficiently large so that

$$\frac{1}{2} \cdot \log_2\left(\frac{4}{3}\right) \cdot (n - 1) > \log_2(\gamma) \text{ (*)}$$

(otherwise we will be able to solve $f - SPP_\gamma$ even without finding a single collision of the hash function). It is necessary that the inequality (*) should be satisfied in order to consider at all the issue of security. Unfortunately

Table 3: Security (collision-resistance) of the hash function used for the Signature Tree against LLL-reduction attacks using 1.18

n	d	m	p	f(X)	ϵ	Hashlength	γ	LLL security
128	7	7	$2^{21.50}$	$X^{128} + 1$	3	2752	$2^{24.4}$	2^{127}
137	8	8	$2^{23.06}$	$X^{136} \dots + X + 1$	6	3160	$2^{26.92}$	2^{136}
256	8	8	$2^{23.59}$	$X^{256} + 1$	3	6040	$2^{26.17}$	2^{251}

it is not sufficient: as we will see later in 4.2, 5.2 in practice and on average the LLL algorithm yields shorter than expected vectors and therefore we will need therefore a bigger value for the parameter n. For more details on the analysis on average and on the choice of values for n in this case see 4.2, 5.2.

The inequality (*) is true for values of n superior to 120 (for polynomials of the form $X^n + 1$) and for values of n superior to 130 (for polynomials of the form $X^n \dots + X + 1$). So in any case we could take only values of n superior to 120. And therefore in the case where

$$0.2075185 \cdot (n - 1) > \log_2(\gamma)$$

we obtain an inequality for the time to find a collision of the hash function:

$$\mathcal{C}(\text{Col}(n)) > \frac{2^n - n^3 \log(n)}{(\lceil 0.2075185 \cdot (n-1) - \log_2(8 \cdot \text{EF}(f,3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2) \rceil)} - m \cdot \left(\frac{2n^3}{3} + n^2 + n^2 \cdot \log(n) \cdot \log \log(n) + 2 \cdot n \right)$$

The choice of $n = 128$ and for the polynomial $X^{128} + 1$ is interesting because we obtain relatively small length of the hash function and sufficient security if the LLL algorithm always yields a vector (almost) equal to the theoretical bound given by theorem 1.16.

Unfortunately in practice and on average the choice of $X^{128} + 1$ and $n = 128$ is insecure and we need to choose $n \geq 500$ (see 4.2).

Here are the necessary formulas for the previous calculus (see for instance p.9 [4]): $d = \lceil \log_2(n) \rceil$, $m = \lceil \log_2(n) \rceil$, $p = \lceil 2 \cdot \epsilon \cdot d \cdot m \cdot n^{1.5} \cdot \log(n) \rceil$, $\text{hashlength} = \log_2 p^n$, $\gamma = 8 \cdot \epsilon^2 \cdot d \cdot m \cdot n \cdot \log^2(n) = 8 \cdot \epsilon^2 \cdot n \cdot \log^4(n)$, $\text{Security} = \frac{2^n - n^3 \log(n)}{(\lceil 0.2075185 \cdot (n-1) - \log_2(8 \cdot \text{EF}(f,3)^2 \cdot d \cdot m \cdot n \cdot \log(n)^2) \rceil)} - m \cdot \left(\frac{2n^3}{3} + n^2 + n^2 \cdot \log(n) \cdot \log \log(n) + 2 \cdot n \right)$.

4.2 Security of the Micciancio-Lyubashevsky's hash function taking into account the results from LLL on the average [42]

It is widely known fact that on the average Lenstra-Lenstra-Lovasz algorithm performs much better than the theoretical bound. See for instance Odlyzko's article [51] p.14 where the author states that the LLL algorithm on average is

Table 4: LLL on average performance and the minimal values of the security parameter n to ensure some security for the Micciancio-Lyubashevsky's hash function

n_{crh}	d	m	p	$f(X)$	ϵ	Hashlength	γ
512	9	9	$2^{25.59}$	$X^{512} + 1$	3	13102	$2^{27.85}$
557	10	10	$2^{27.10}$	$X^{556} \dots + X + 1$	6	15094	$2^{30.31}$
563	10	10	$2^{27.13}$	$X^{562} \dots + X + 1$	6	15274	$2^{30.33}$

much better than expected and even for small dimensions the LLL algorithm finds the shortest non-zero vector in a lattice.

In [42] Phong Nguyen and Damien Stehlé proposed a modelization of the average behaviour of the LLL algorithm, changing the constant $\frac{4}{3}$ from the initial formulation of LLL 1.16 by 1.08. In the following calculus we use the formulation from Nguyen and Stehlé:

$$(1.08)^{\frac{n-1}{2}} > 8 \cdot \epsilon^2 \cdot n \cdot \log^4(n)$$

which is equivalent to:

$$0.0555 \cdot (n - 1) > \log_2(8 \cdot \epsilon^2 \cdot n \cdot \log^4(n))$$

Solving this inequality for n gives us:

$$n \geq 500 \text{ if } \epsilon = 3$$

and

$$n \geq 550 \text{ if } \epsilon = 6$$

In table 4 are given some possible values for the parameter n_{crh} and the corresponding polynomials:

Remark. We didn't give explicit values of the security of the LLL on average, because this result is about an on average performance of an algorithm and not about a theoretical performance. Moreover as we will see in the next paragraphs 4.3, 4.4 the values of n in the interval $600 \geq n \geq 500$ does not provide enough security against attacks using the RSR-reduction 1.19 or the primal-dual RSR reduction 1.21.

Table 4 has the goal to illustrate an interesting aspect of the LLL algorithm: the vectors obtained by the application of this algorithm are much shorter than suggested by the theoretical bound. Consequently the parameter n_{crh} that we have chosen in the previous paragraph so that

$$\frac{1}{2} \cdot \log_2\left(\frac{4}{3}\right) \cdot (n_{crh} - 1) \gtrsim \log_2(\gamma)$$

(paragraph 4.1, table 3) should be in practice on average much larger. So as we already remarked in 4.1, the choice of $X^{128} + 1$ ($n_{\text{crh}} = 128$) (see table 3) is not secure enough in practice and we should choose a polynomial with larger degree such as: $X^{512} + 1$ ($n_{\text{crh}} = 512$). See table 4 for more polynomials.

Remark. The study of LLL on the average is not only important for the pure LLL-reductions, but it is also important for the BKZ-reductions 1.19, 1.20, 1.21 because the BKZ-reductions use the LLL-reduction in their algorithm (see [42], [41]).

4.3 Security of the Micciancio-Lyubashevsky's hash function (used for hashing the Merkle tree) when the attacker use the Schnorr's RSR algorithm 1.19 (a BKZ lattice reduction algorithm)

For the estimation of the security of the collision-resistance of the hash function we suppose again that:

$$T(n_{\text{crh}}) \geq \mathcal{C}(f - SPP_{\mathcal{O}(n^4)}) \Rightarrow T(n_{\text{crh}}) \geq 2^{n_{\text{crh}}}$$

which is equivalent in this case to:

$$\mathcal{O}(n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} + n^4) + \lceil \frac{n}{2k} \cdot \log_2 \left(\frac{k}{6}\right) - \log_2(\gamma) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3}) + n^2 \cdot \log(n) \cdot \log \log(n) + n) + \mathcal{C}(\text{Col}(n)) \geq 2^n$$

First of all we should choose n and k so that the time for the RSR-reduction $\mathcal{O}(n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} + n^4)$ is feasible. That means we should impose:

$$\mathcal{O}(n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} + n^4) \lesssim 2^{80+r}$$

where $r \geq 0$ is a positive number, which we will use to estimate the security of the hash function.

In particular we should have:

$$n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{80+r}$$

We will first suppose $r = 0$ and prove that the hash function will be secure against attacks with today's computer power. Then in a more detailed analysis we will suppose $r > 0$ and find the exact expression for the security of the hash function.

4.3.1 1st case: $r = 0$

If $r = 0$ we obtain immediately the inequality:

$$n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{80}$$

As $n \geq 128 = 2^7$ (otherwise we would be able to solve the $f - SPP_\gamma$ -problem with the Schnorr's variation of the LLL-algorithm considered in the previous section 4.1) we obtain:

$$\left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{59}$$

i.e.

$$k \leq 70$$

On the other hand we should have $k \geq 24$ so the parameter k satisfies

$$70 \geq k \geq 24$$

We should have:

$$\frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) > \log_2(\gamma)$$

i.e.

$$\frac{1}{2k} \cdot \log_2\left(\frac{k}{6}\right) > \frac{1}{n} \log_2(8 \cdot \epsilon^2 \cdot n \cdot \log^4(n))$$

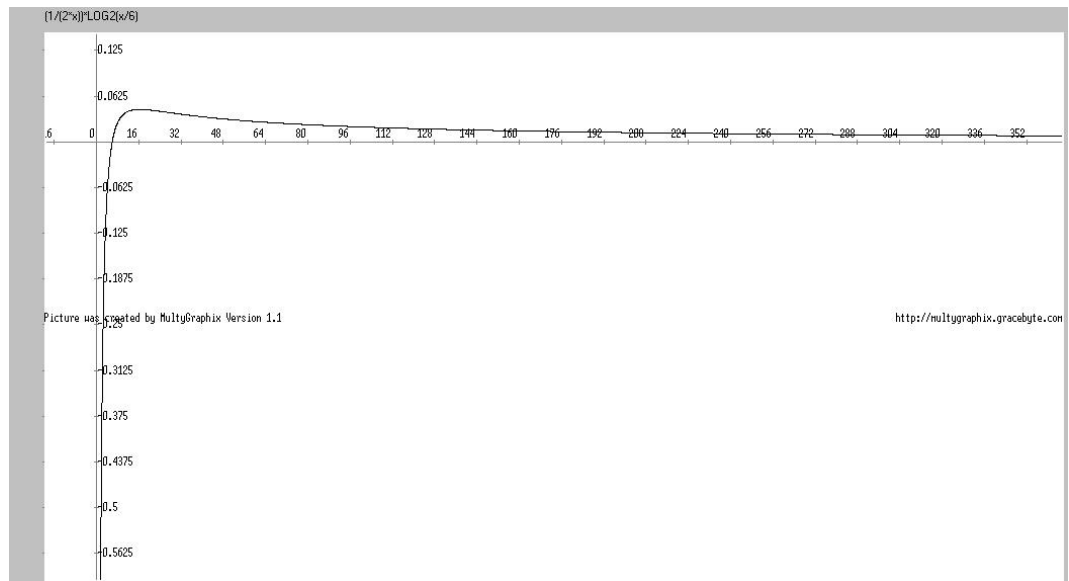


Figure 4: The plot of the function $\frac{1}{2x} \cdot \log_2\left(\frac{x}{6}\right)$

The function $f(x) = \frac{1}{2x} \cdot \log_2\left(\frac{x}{6}\right)$ has a first derivative equal to:

$$f'(x) = -\frac{1}{2x^2} \cdot \log_2\left(\frac{x}{6}\right) + \frac{1}{2x \cdot \ln(2) \cdot 6} \cdot \frac{6}{x} = \frac{1}{2x^2 \ln 2} \cdot (1 - \ln\left(\frac{x}{6}\right))$$

$$f'(x) = 0 \Rightarrow x = 6 \cdot e$$

So there is an extremum (maximum) of the function $f(x)$ for

$$x = 6 \cdot e \approx 16.31$$

The function $f(x)$ is decreasing in the interval $[16.31, \infty]$ and therefore decreasing also in the interval $[24, 70]$.

The minimal value of $f(x)$ in this interval is therefore $f(70) = 0.0253165$. We should find a constant c so that 0.0253165 is superior to $g(n) = \frac{1}{n} \log_2(8 \cdot \epsilon^2 \cdot n \cdot \log^4(n))$ for all $n > c$. Such a constant is for instance $c = 1200$ because $g(n)$ is a decreasing function and $0.0253165 > g(1200)$, (so $\forall n > 1200$, $0.0253165 > g(n)$). For these values of n the approximation factor obtained by RSR algorithm will be superior to the approximation factor γ in SPP_γ and in order to find a solution to SPP_γ with $\gamma \lesssim \tilde{O}(n^4)$ we will not have other choice than apply the general algorithm proposed by Kumar and Sivakumar (see Preliminaries 1.17).

Remark about the values of k : If we make a more detailed analysis of the inequality $n^3(\frac{k}{6})^{\frac{k}{4}} \lesssim 2^{80}$ and take into account that the values of n that we will consider will be ≥ 1200 we will obtain that the inequality that k will satisfy in this case will be $60 \geq k \geq 24$. For the rest of the subsection we will consider that $60 \geq k$.

So we have the inequality:

$$\mathcal{O}(n^3(\frac{k}{6})^{\frac{k}{4}} + n^4) + \lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil \cdot (m \cdot (n^2 + n + \frac{2n^3}{3} + n^2 \cdot \log(n) \cdot \log \log(n) + n) + \mathcal{C}(\text{Col}(n))) \geq 2^n$$

and

$$\mathcal{C}(\text{Col}(n)) \gtrsim \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil)}$$

so the security of the hash function is:

$$\text{Security} = \min\{2^{80}, \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil)}\}$$

The values for $\frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil)}$ that we obtain respectively for $n = 1201, n = 1213, n = 2024$ (and the polynomials in the following table) are $2^{1199}, 2^{1211}, 2^{2044}$

So we will have in all these cases:

$$\text{Security} \geq 2^{80}$$

i.e. Micciancio-Lyubashevsky's hash function for $n \geq 1200$ will be secure against today's computer power.

Here are the necessary formulas for the previous calculus (see for instance p.9 [4]): $d = \lceil \log_2(n) \rceil$, $m = \lceil \log_2(n) \rceil$, $p = \lceil 2 \cdot \epsilon \cdot d \cdot m \cdot n^{1.5} \cdot \log(n) \rceil$, $\text{hashlength} = \log_2(p^n)$, $\gamma = 8 \cdot \epsilon^2 \cdot d \cdot m \cdot n \cdot \log^2(n) = 8 \cdot \epsilon^2 \cdot n \cdot \log^4(n)$.

Table 5: Security (collision-resistance) of the hash function against attacks with today's computer's power, used for the Signature Tree for different values of the security parameter n against RSR-attacks

n	d	m	p	f(X)	ϵ	Hashlength	γ	Security
1201	11	11	$2^{29.20}$	$X^{1200} + \dots + X + 1$	6	35070	$2^{32.03}$	security $\geq 2^{80}$
1213	11	11	$2^{29.23}$	$X^{1212} \dots + X + 1$	6	35456	$2^{32.05}$	security $\geq 2^{80}$
2048	11	11	$2^{27.88}$	$X^{2048} + 1$	3	57098	$2^{31.01}$	security $\geq 2^{80}$

4.3.2 2^{nd} case : $r > 0$

We have the equality:

$$n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} = 2^{80+r}$$

and $n > 1024$ so we obtain:

$$\left(\frac{k}{6}\right)^{\frac{k}{4}} = 2^{50+r}$$

and therefore:

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 50$$

So k is an implicit function of r .

Moreover we have for the security of the SPP_γ -problem:

$$\text{Security} = \min\left\{2^{80+r}, \frac{2^n}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)}\right\}$$

For a fixed value of n we can maximize the security if we maximize the minimum of the two quantities.

This can happen when:

$$2^{80+r} = \frac{2^n}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)}$$

So taking into account that at the limit $\left(\frac{k}{6}\right)^{\frac{k}{4}} = 2^{50+r}$ we obtain:

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 50$$

$$2^{80+\frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 50} = \frac{2^n}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)}$$

$$2^{30+\frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)} = \frac{2^n}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)}$$

$$n = 30 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) + \log\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)$$

We can neglect every term in $\log\log(n)$ (for $n \approx 1200$, $\log\log(n) < 4$) so we obtain:

$$n \approx 30 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) + \log(n) - \log(2k)$$

Moreover in the case of the RSR reduction (see [37]) the following inequality should be satisfied by the parameter k :

$$k \cdot (3 + \log_2(k)) \leq n$$

As the values of n that we consider are $n \geq 1200$ and as $n \approx 30 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) + \log(n) - \log(2k)$, the predominant term on the right-hand side will be $\frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)$ it turns out $n \approx \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)$ but for these values of k the inequality $k \cdot (3 + \log_2(k)) \leq n$ is not satisfied.

So what will limit the value of k will not be the equation:

$$2^{80+r} = \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil)}$$

but rather the inequality $k \cdot (3 + \log_2(k)) \leq n$.

For these values of k we will have:

$$2^{80+r} < \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil)}$$

and

$$\text{Security of the hash function} = 2^{80+r}$$

where r is such that:

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 50$$

and where $k \cdot (3 + \log_2(k)) = n$

If we denote the $g(x) = x \cdot (3 + \log_2(x))$ we have:

$$k = g^{-1}(n)$$

$$\text{Security of the hash function} = 2^{30 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)}$$

and if we replace k by $g^{-1}(n)$ we obtain:

$$\text{Security of the hash function} = 2^{(30 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{6}\right))}$$

We can now calculate the exact security for the polynomials from the case $r = 0$:

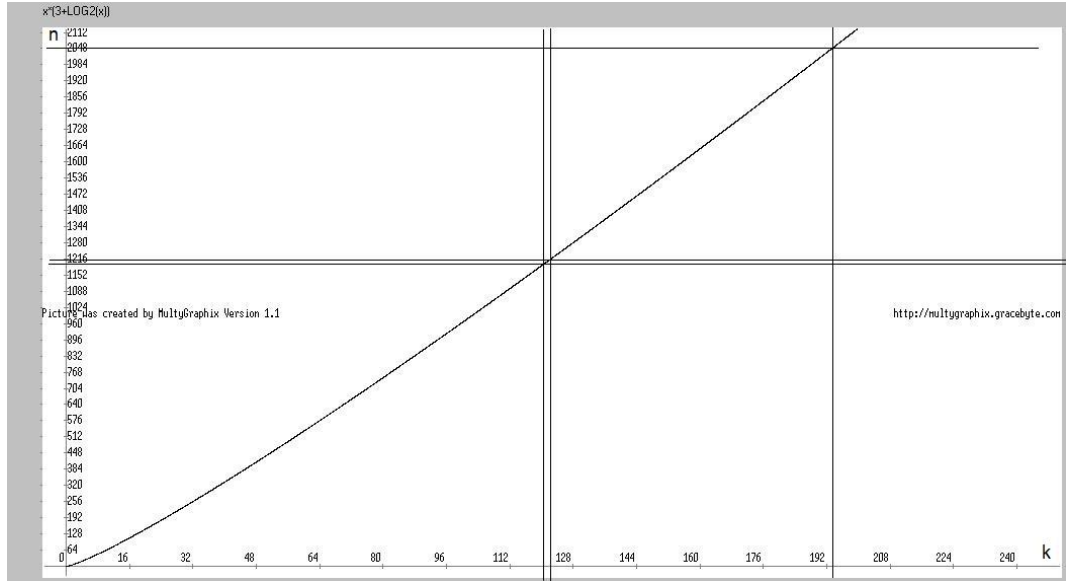


Figure 5: The plot of the function $g(x) = x \cdot (3 + \log_2(x))$. In abscissa are the values of k , and in ordinate are the respective values of n . For $n = 1201$, $k = 121$; for $n = 1213$, $k = 122$ and for $n = 2048$ we have $k = 194$.

Table 6: Exact Security (collision-resistance) of the hash function used for the Signature Tree for different values of the security parameter n against RSR-attacks

n	d	m	p	f(X)	ϵ	Hashlength	γ	Security =
1201	11	11	$2^{29.20}$	$X^{1200} + \dots + X + 1$	6	35070	$2^{32.03}$	2^{161}
1213	11	11	$2^{29.23}$	$X^{1212} \dots + X + 1$	6	35456	$2^{32.05}$	2^{163}
2048	11	11	$2^{27.88}$	$X^{2048} + 1$	3	57098	$2^{31.01}$	2^{273}

4.4 Security of the Micciancio-Lyubashevsky's hash function (used for hashing the Merkle tree) when the attacker uses the primal-dual RSR algorithm 1.21 (a BKZ lattice reduction algorithm)

We should have the inequality:

$$(1.025)^{\frac{n}{2}} > \gamma_{\text{crh}}$$

and therefore

$$(1.025)^{\frac{n}{2}} > 8 \cdot \epsilon^2 \cdot n \cdot \log^4(n)$$

which is equivalent to:

$$0.01781 \cdot n > \log_2(8 \cdot \epsilon^2 \cdot n \cdot \log^4(n))$$

Table 7: Exact Security (collision-resistance) of the hash function used for the Signature Tree for different values of the security parameter n against primal-dual RSR -attacks for $k = 80$

n	d	m	p	f(X)	ϵ	Hashlength	γ	Security =
1847	11	11	$2^{30.22}$	$X^{1846} + \dots + X + 1$	6	55816	$2^{32.78}$	$\geq 2^{240}$
1861	11	11	$2^{30.24}$	$X^{1860} \dots + X + 1$	6	56277	$2^{32.80}$	$\geq 2^{243}$
2048	11	11	$2^{27.88}$	$X^{2048} + 1$	3	57098	$2^{31.01}$	$\geq 2^{268}$

which is satisfied for:

$$n \geq 1720 \text{ (in the case } \epsilon = 3)$$

and for

$$n \geq 1840 \text{ (in the case } \epsilon = 6)$$

In the same way as in the paragraph (case $r > 0$) 4.3.2 we obtain that the security of the hash function is:

$$\text{Security} = \min\left\{2^{80+r}, \frac{C(f-SPP_\gamma)}{(\lceil \frac{n}{2^k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}\right\}$$

and if we suppose that the complexity of the SPP_γ with $\gamma \lesssim n^4$ is 2^n (see 1.17, 3.3) we obtain:

$$\text{Security} = \min\left\{2^{80+r}, \frac{2^n}{(\lceil \frac{n}{2^k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}\right\}$$

In the same way as in (case $r > 0$) 4.3.2 we obtain that for $n > 1700$:

$$2^{80+r} < \frac{2^n}{(\lceil \frac{n}{2^k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}$$

and therefore:

$$\text{Security of the hash function} = 2^{80+r}$$

where r is such that:

$$n^2 \cdot \log_2(n) \cdot \left(\frac{k}{11}\right)^{\frac{k}{4}} = 2^{80+r}$$

and taking $n \geq 1720$ we obtain for r :

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{11}\right) - 55$$

where $k \cdot (3 + \log_2(k)) = n$ (We suppose here that the inequality $k \cdot (3 + \log_2(k)) \leq n$ is true in the case of primal-dual RSR, in the same way as it was true in the case of RSR)

If we denote the $g(x) = x \cdot (3 + \log_2(x))$ we have:

$$k = g^{-1}(n)$$

and therefore the security of the hash function is:

$$\text{Security of the hash function} = 2^{25 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{11}\right)}$$

Respectively for $n = 1847, 1861$ and 2048 we obtain security of $2^{201}, 2^{204}, 2^{226}$.

4.5 Conclusion of this section

In paragraph 4.1 we found the minimum values of the security parameter ($n_{\text{crh}} \geq 128$) which could make the hash function secure against LLL-type attacks.

Unfortunately in practice and on average the LLL algorithm performs much better than the theoretical bound suggests and we need much bigger values for the security parameter ($n_{\text{crh}} \geq 512$) in order to ensure the security on average against LLL attacks. See 4.2 and [42].

In paragraph 4.3 we used a quite efficient algorithm called RSR proposed by Claus Peter Schnorr 1.19 which makes the polynomials of degree inferior to 1200 insecure under the RA and GSA assumptions. So $n_{\text{crh}} \geq 1200$ is a necessary (but not sufficient) condition for security of the Micciancio-Lyubashevsky's hash function.

In paragraph 4.4 we used the algorithm from 1.21 which in feasible time and on average produce a very good (but still an exponential) approximation of the shortest vector. In this paragraph we obtain that in order to consider security against an attack using this algorithm we should impose $n_{\text{crh}} \geq 1840$.

For estimating the security of the hash function and giving some concrete values we made the following assumption:

$$\mathcal{C}(f - SPP_\gamma \text{ with } \gamma = \mathcal{O}(n^4)) \approx 2^n \quad (*)$$

which is supposedly true for sufficiently large values of n .

In the cases where we have parameterized reduction (for example RSR-reduction, primal-dual-reduction ...) we have the following general expression of the security:

Security against primal-dual RSR 1.21 reduction attack

$$\text{Security} = \min\left\{2^{25 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{11}\right)}, \frac{\mathcal{C}(f - SPP_\gamma)}{\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil}\right\}$$

Security against RSR 1.19 reduction attack

$$\text{Security} = \min\left\{2^{(30 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{6}\right))}, \frac{\mathcal{C}(f - SPP_\gamma)}{\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil}\right\}$$

with $g(n) = n \cdot (3 + \log_2(n))$.

which only the assumption (*) simplifies to:

Simplified security against primal-dual RSR 1.21 reduction attack

$$\text{Security} = 2^{25 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{11}\right)}$$

Table 8: A resume of section 4

Algorithm	$n_{\text{crh}} \geq$	Hashlength \geq	Security
LLL 1.18	128	2750	2^{124}
LLL (average) 4.2	512	13100	-
RSR 1.19	1200	35050	2^{161}
Primal-dual RSR 1.21	1840	55800	2^{240}

The reduction algorithms and the corresponding minimal values of the security parameters and the hash length are given in the following table:

The security of the Micciancio-Lyubashevsky's hash should be the security of hash function against the most powerful of the lattice reductions i.e. 2^{240} for $X^{1846} + \dots + 1$ against the primal-dual RSR 1.21. Unfortunately this algorithm makes some assumptions such as RA and GSA, and more importantly has a quite significant time complexity which makes that sometimes the reduction is not feasible in practice. That is why the results about some other algorithms (see table 8) could be also of interest.

5 Security of the Micciancio-Lyubashevsky's one-time signature

The security of the Micciancio-Lyubashevsky's one-Time Signature [7] is based on the difficulty of finding a collision for the underlying hash function (see Theorem 1.23 (Preliminaries)) which in turn is based on the difficulty of finding a solution to the $f - SPP_\gamma$ problem for some small value of the approximation factor γ (see Theorem 1.27 (Preliminaries)). As we have already considered similar problem in the previous section we can use many of the formulas that we already obtained, only we should take into account that the approximation factor γ is no more equal to $8 \cdot \epsilon^2 \cdot d \cdot m \cdot n \cdot \log^2(n)$, but instead equal to $80 \cdot \phi^3 \cdot n^2(\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n)$.

5.1 Security of Micciancio-Lyubashevsky's one-Time Signature [7] when the attacker use a LLL type reduction algorithm 1.18

We should again suppose that the approximation factor obtained after the application of the improved LLL algorithm is larger than the approximation factor γ that we want to obtain:

$$\frac{1}{2} \cdot \log_2\left(\frac{4}{3}\right) \cdot (n - 1) > \log_2(\gamma)$$

Using the expression of γ as a function of n and giving different values for n we obtain that the previous inequality is true for $n > 205$. We will consider therefore only values of n superior to 205 in this subsection.

Table 9: LLL security of the one-time signature for different values of the security parameter n . One should pay attention to the fact that when choose a polynomial of the form $X^{n-1} \dots + X + 1$ n should be prime and when we choose polynomials of the form $X^n + 1$ n should be a power of 2

n	d	m	p	f(X)	ϕ	Hashlength	γ	LLL security
211	$2^{21.2}$	8	$2^{26.16}$	$X^{210} + \dots + X + 1$	2	5520	$2^{42.78}$	2^{211}
223	$2^{21.35}$	8	$2^{26.40}$	$X^{222} + \dots + X + 1$	2	5887	$2^{43.04}$	2^{221}
256	$2^{20.32}$	8	$2^{24.00}$	$X^{256} + 1$	1	6144	$2^{40.32}$	2^{251}

We can then calculate the security using the formula from the previous subsection:

$$\mathcal{C}(\text{Col}(n)) > \frac{2^n - n^3 \log(n)}{\lceil 0.2075185 \cdot (n-1) - \log_2(80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n)) \rceil} - \log(n) \cdot \left(\frac{2n^3}{3} + n^2 + n^2 \cdot \log(n) \cdot \log \log(n) + 2 \cdot n\right)$$

For different different choices of the polynomial function we obtain the following table:

The choice of $n = 211$ and of the polynomial $X^{210} + \dots + X + 1$ is interesting because we obtain relatively small length of the hash function and sufficient security if the LLL algorithm always yields a vector (almost) equal to the theoretical bound given by theorem 1.16. Unfortunately in practice and on average the choice of $X^{210} + \dots + X + 1$ and $n = 211$ is insecure and we need to choose $n \geq 790$ (see 5.2).

Here are some of the formulas that we used for the calculus: $d = 10\phi p^{1/m} n \log^2(n)$, $m = \lceil \log_2(n) \rceil$, $p = (\phi n)^3$, $\text{hashlength} = \log p^n$, $\gamma = 80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n)$, $\text{Security} = \frac{2^n - n^3 \log(n)}{\lceil 0.2075185 \cdot (n-1) - \log_2(80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n)) \rceil} - \log(n) \cdot \left(\frac{2n^3}{3} + n^2 + n^2 \cdot \log(n) \cdot \log \log(n) + 2 \cdot n\right)$.

5.2 Security of the Micciancio-Lyubashevsky's one-time signature when we take into account the results from LLL on the average [42]

It is widely known fact that on the average Lenstra-Lenstra-Lovasz algorithm performs much better than the theoretical bound. See for instance Odlyzko's article [51] p.14 where the author states that the LLL algorithm on average is much better than expected and even for small dimensions the LLL algorithm finds the shortest non-zero vector in a lattice.

As in paragraph 4.2 we change the constant $\frac{4}{3}$ with 1.08:

$$(1.08)^{\frac{n-1}{2}} > 80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n)$$

which is equivalent to:

$$0.0555 \cdot (n-1) > \log_2(80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n))$$

Table 10: LLL on average security of the one-time signature for different values of the security parameter n

n	d	m	p	$f(X)$	ϵ	Hashlength	γ
907	$2^{23.99}$	10	$2^{32.47}$	$X^{906} + \dots + X + 1$	2	29450	$2^{48.70}$
911	$2^{24.00}$	10	$2^{32.49}$	$X^{910} \dots + X + 1$	2	29598	$2^{48.72}$
1024	$2^{22.97}$	10	2^{30}	$X^{1024} + 1$	1	30720	$2^{45.93}$

Solving this inequality for n gives us:

$$n \geq 790 \text{ if } \phi = 1$$

and

$$n \geq 890 \text{ if } \epsilon = 2$$

Remark. We didn't give explicit values of the security of the LLL on average, because as we will see in the next paragraphs 5.3, 5.4 the values of n in the interval $1100 \geq n \geq 900$ does not provide security against attacks using the RSR-reduction 1.19 or the primal-dual RSR reduction 1.21.

Remark. Table 10 illustrates an interesting aspect of the LLL algorithm: the vectors obtained by the application of this algorithm are much shorter than suggested by the theoretical bound. Consequently the parameter n_{crh} that we have chosen in the previous paragraph so that

$$\frac{1}{2} \cdot \log_2\left(\frac{4}{3}\right) \cdot (n_{\text{crh}} - 1) \gtrsim \log_2(\gamma)$$

(paragraph 5.1, table 9) should be in practice on average much larger. So as we already remarked in 5.1, the choice of $n = 211$ and $X^{210} + \dots + X + 1$ (see table 9) is not secure enough in practice and we should choose a polynomial with larger degree such as: $X^{906} + \dots + X + 1$ ($n_{\text{crh}} = 907$). See table 10 for more polynomials.

Remark. The study of LLL on the average is not only important for the pure LLL-reductions, but it is also for BKZ-reductions 1.19, 1.20, 1.21 because the BKZ-reductions use the LLL-reduction in their algorithm (see [41], [42]).

5.3 Security of Micciancio-Lyubashevsky's one-Time Signature [7] when the attacker use the algorithm RSR Schnorr 1.19 (a BKZ lattice reduction algorithm)

We suppose again as in paragraph 4.3 that:

$$n^3 \cdot \left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{80+r}$$

where $r > 0$ is a positive number.

Remark: The reason for this assumption is that the RSR algorithm should be able to finish its execution for the chosen values of n and k .

We consider the same cases as in the paragraph 4.3: $r = 0$ and $r > 0$:

5.3.1 1st case: $r = 0$

If $r = 0$ we obtain:

$$n^3 \cdot \left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{80}$$

As $n > 210$ (see paragraph 5.1) and using the previous inequality we have that:

$$\left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{57}$$

i.e.

$$k \leq 66$$

As on the other hand we have that $k \geq 24$ we obtain:

$$66 \geq k \geq 24$$

As in paragraph 4.3 we should have:

$$\frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) > \log_2(\gamma)$$

(otherwise we will only need one application of the RSR-algorithm which will solve our problem)

We define as in paragraph 4.3 the function $f(x) = \frac{1}{2x} \cdot \log_2\left(\frac{x}{6}\right)$ which has a maximum in $x = 6 \cdot e = 16.31$ and which is decreasing in the interval $[24, 66]$. The minimal value of $f(x)$ in this interval will be therefore $f(66) = 0.02621$.

We should find a constant c such that 0.02621 is superior to $g(n) = \frac{1}{n} \log_2(80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n))$ for $\forall n > c$. Such a constant is for instance $c = 1950$ because $g(n)$ is a decreasing function and $0.02621 > g(1950)$. For these values of n the approximation factor obtained by RSR algorithm will be superior to the approximation factor γ in $f - SPP_\gamma$ and in order to find a solution to SPP_γ with $\gamma \lesssim \tilde{O}(n^4)$ we will not have other choice than apply the general algorithm proposed by Kumar and Sivakumar (see Preliminaries).

Remark about the values of k : If we make a more detailed analysis of the inequality $n^3 \cdot \left(\frac{k}{6}\right)^{\frac{k}{4}} \lesssim 2^{80}$ and take into account that the values of n that we will consider will be ≥ 1950 we will obtain that the inequality that k will satisfy in this case will be $48 \geq k \geq 24$. For the rest of the subsection we will consider that $48 \geq k$.

So we have the inequality:

$$\mathcal{O}\left(n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} + n^4\right) + \left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma) \rceil\right) \cdot \left(m \cdot (n^2 + n + \frac{2n^3}{3}) + n^2 \cdot \log(n) \cdot \log \log(n) + n\right) + \mathcal{C}(\text{Col}(n)) \geq 2^n$$

and

$$\mathcal{C}(\text{Col}(n)) \gtrsim \frac{2^n}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma) \rceil\right)}$$

so the security of the hash function is:

$$\text{Security} = \min\left\{2^{80}, \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil)}\right\}$$

For $n > 1950$ we have $\frac{2^n}{\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma) \rceil} \gg 2^{80}$ and so we obtain:

$$\text{Security} \geq 2^{80}$$

i.e. Micciancio-Lyubashevsky's hash function for $n \geq 1200$ will be secure against today's computer power.

5.3.2 2nd case: $r > 0$

We have the equality:

$$n^3 \left(\frac{k}{6}\right)^{\frac{k}{4}} = 2^{80+r}$$

and also using $n > 1950$ we obtain:

$$\left(\frac{k}{6}\right)^{\frac{k}{4}} = 2^{48+r}$$

and therefore:

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 48$$

So k is an implicit function of r .

Moreover we have for the security of the SPP_γ -problem:

$$\text{Security} = \min\left\{2^{80+r}, \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}\right\}$$

For a fixed value of n we can maximize the security if we maximize the minimum of the two quantities.

This can happen when:

$$2^{80+r} = \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}$$

So taking into account that at the limit $\left(\frac{k}{6}\right)^{\frac{k}{4}} = 2^{48+r}$ we obtain:

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 48$$

$$2^{80 + \frac{k}{4} \cdot \log_2(\frac{k}{6}) - 48} = \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}$$

$$2^{32 + \frac{k}{4} \cdot \log_2(\frac{k}{6})} = \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}$$

$$n = 32 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) + \log\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)$$

We can neglect every term in $\log\log(n)$ so we obtain:

$$n \approx 30 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) + \log(n) - \log(2k)$$

Moreover in the case of the RSR reduction (see [37]) the following inequality should be satisfied by the parameter k :

$$k \cdot (3 + \log_2(k)) \leq n$$

As the values of n that we consider are $n \geq 1950$ and as $n \approx 32 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) + \log(n) - \log(2k)$, the predominant term on the right-hand side will be $\frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)$ it turns out $n \approx \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)$ but for these values of k the inequality $k \cdot (3 + \log_2(k)) \leq n$ is not satisfied.

So what will limit the value of k will not be the equation:

$$2^{80+r} = \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil)}$$

but rather the inequality $k \cdot (3 + \log_2(k)) \leq n$.

For these values of k we will have:

$$2^{80+r} < \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil)}$$

and therefore we will have:

$$\text{Security of the hash function} = 2^{80+r}$$

where r is such that:

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right) - 48$$

and where $k \cdot (3 + \log_2(k)) = n$

If we denote the $g(x) = x \cdot (3 + \log_2(x))$ we have:

$$k = g^{-1}(n)$$

$$\text{Security of the hash function} = 2^{32 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)}$$

and if we replace k by $g^{-1}(n)$ we obtain:

$$\text{Security of the hash function} = 2^{(32 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{6}\right))}$$

We can now calculate the exact security for some polynomials with degree n such that $n \geq 1950$:

Here are some of the formulas that we used for the calculus: $d = 10\phi p^{1/m} n \log^2(n)$, $m = \lceil \log_2(n) \rceil$, $p = (\phi n)^3$, $\text{hashlength} = \log p^n$, $\gamma = 80 \cdot \phi^3 \cdot n^2 (\phi n)^{3/\lceil \log_2(n) \rceil} \log_2^5(n)$, $\text{Security} = 2^{(32 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{6}\right))}$.

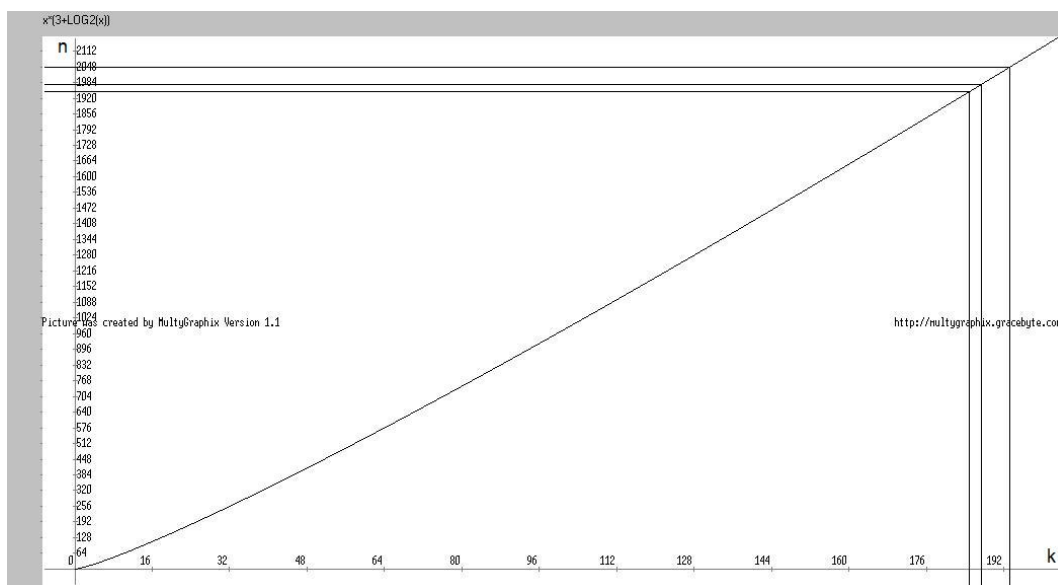


Figure 6: The plot of the function $g(x) = x \cdot (3 + \log_2(x))$. In abscissa are the values of k , and in ordinate are the respective values of n . For $n = 1951$, $k = 185$; for $n = 1973$, $k = 187$ and for $n = 2048$ we have $k = 194$.

Table 11: Exact Security of the one-time signature for different values of the security parameter n against RSR-attacks

n	d	m	p	f(X)	ϕ	Hashlength	γ	Security =
1951	$2^{25.41}$	11	$2^{35.79}$	$X^{1950} + \dots + X + 1$	2	69826	$2^{51.69}$	2^{261}
1973	$2^{25.43}$	11	$2^{35.84}$	$X^{1972} \dots + X + 1$	2	70712	$2^{51.73}$	2^{264}
2048	$2^{24.24}$	11	$2^{33.00}$	$X^{2048} + 1$	1	67584	$2^{48.62}$	2^{275}

5.4 Security of the Micciancio-Lyubashevsky's one-time signature when the attacker uses the primal-dual RSR algorithm 1.21 (a BKZ lattice reduction algorithm)

We should have the inequality:

$$(1.025)^{\frac{n_{\text{ots}}}{2}} > \gamma_{\text{ots}}$$

and therefore

$$(1.025)^{\frac{n_{\text{ots}}}{2}} > 80 \cdot \phi^3 \cdot n_{\text{ots}}^2 (\phi n_{\text{ots}})^{3/\lceil \log_2(n_{\text{ots}}) \rceil} \log_2^5(n_{\text{ots}})$$

which is equivalent to:

$$0.01781 \cdot n_{\text{ots}} > \log_2(80 \cdot \phi^3 \cdot n_{\text{ots}}^2 (\phi n_{\text{ots}})^{3/\lceil \log_2(n_{\text{ots}}) \rceil} \log_2^5(n_{\text{ots}}))$$

which is satisfied for:

Table 12: Exact Security of the Micciancio-Lyubashevsky's one-time signature for different values of the security parameter n against primal-dual RSR -attacks for $k = 80$

n	d	m	p	f(X)	ϵ	Hashlength	γ	Security
3001	$2^{26.07}$	12	$2^{37.65}$	$X^{3000} + \dots + X + 1$	2	78236	$2^{53.21}$	$\geq 2^{340}$
3011	$2^{26.08}$	12	$2^{37.67}$	$X^{3010} \dots + X + 1$	2	78526	$2^{53.23}$	$\geq 2^{342}$
4096	$2^{25.49}$	12	$2^{36.00}$	$X^{4096} + 1$	1	104415	$2^{51.25}$	$\geq 2^{475}$

$$n_{\text{ots}} \geq 2800 \text{ (in the case } \phi = 1)$$

and for

$$n_{\text{ots}} \geq 3000 \text{ (in the case } \phi = 2)$$

In the same way as in the paragraph (case $r > 0$) 5.3.2 we obtain that the security of the one-time signature is:

$$\text{Security} = \min\left\{2^{80+r}, \frac{\mathcal{C}(f-SPP_\gamma)}{\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil}\right\}$$

and if we suppose that the complexity of the SPP_γ with $\gamma \lesssim n^4$ is 2^n (see 1.17, 3.3) we obtain:

$$\text{Security} = \min\left\{2^{80+r}, \frac{2^n}{\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil}\right\}$$

In the same way as in (case $r > 0$) 5.3.2 we obtain that for $n > 3000$:

$$2^{80+r} < \frac{2^n}{(\lceil \frac{n}{2k} \cdot \log_2(\frac{k}{6}) - \log_2(\gamma(n)) \rceil)}$$

and therefore:

$$\text{Security of the one-time signature} = 2^{80+r}$$

where r is such that:

$$n^2 \cdot \log_2(n) \cdot \left(\frac{k}{11}\right)^{\frac{k}{4}} = 2^{80+r}$$

and taking $n \geq 3000$ we obtain for r :

$$r = \frac{k}{4} \cdot \log_2\left(\frac{k}{11}\right) - 53$$

where $k \cdot (3 + \log_2(k)) = n$ (We suppose here that the inequality $k \cdot (3 + \log_2(k)) \leq n$ is true in the case of primal-dual RSR, in the same way as it was true in the case of the RSR algorithm).

If we denote $g(x) = x \cdot (3 + \log_2(x))$ we have:

$$k = g^{-1}(n)$$

and therefore the security of the one-time signature is:

$$\text{Security of the one-time signature} = 2^{27 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{11}\right)}$$

Respectively for $n = 3001, 3011$ and 4096 we obtain security of $2^{340}, 2^{342}, 2^{475}$.

5.5 Conclusion of this section

In paragraph 5.1 we found the minimum values of the security parameter ($n_{\text{crh}} \geq 210$) which could make the Micciancio-Lyubashevsky one-time signature secure against LLL-type attacks.

Unfortunately in practice and on average the LLL algorithm performs much better than the theoretical bound suggests and we need much bigger values for the security parameter ($n_{\text{crh}} \geq 890$) in order to ensure the security on average against LLL attacks. See 5.2 and [42].

In paragraph 5.3 we used a quite efficient algorithm called RSR proposed by Claus Peter Schnorr 1.19 which makes the polynomials of degree inferior to 1950 insecure under RA and GSA assumptions. So $n_{\text{crh}} \geq 1950$ is a necessary (but not sufficient) condition for security of the Micciancio-Lyubashevsky's one-time signature.

In paragraph 5.4 we used the algorithm from 1.21 which in almost feasible time on average produces a very good approximation (but still an exponential one) of the shortest vector. In this paragraph we obtain that in order to consider security against an attack using this algorithm we should impose $n_{\text{crh}} \geq 3000$.

For estimating the security of the hash function and giving some concrete values we made the following assumption:

$$\mathcal{C}(f - SPP_{\mathcal{O}(n^4)}) \approx 2^n \quad (1)$$

which is supposedly true for sufficiently large values of n .

In the cases where we have parameterized reduction (for example RSR-reduction, primal-dual-reduction ...) we have the following general expression of the security:

Security of MLots against primal-dual RSR-reductions 1.21

$$\text{Security} = \min\left\{2^{27 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{11}\right)}, \frac{\mathcal{C}(f - SPP_\gamma)}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)}\right\}$$

Security of MLots against RSR-reductions 1.19

$$\text{Security} = \min\left\{2^{32 + \frac{k}{4} \cdot \log_2\left(\frac{k}{6}\right)}, \frac{\mathcal{C}(f - SPP_\gamma)}{\left(\lceil \frac{n}{2k} \cdot \log_2\left(\frac{k}{6}\right) - \log_2(\gamma(n)) \rceil\right)}\right\}$$

with $g(n) = n \cdot (3 + \log_2(n))$.

which under the assumption 1 gives:

Simplified security of MLots against primal-dual RSR-reductions 1.21

$$\text{Security} = 2^{27 + \frac{g^{-1}(n)}{4} \cdot \log_2\left(\frac{g^{-1}(n)}{11}\right)}$$

Table 13: A resume of section 5

Algorithm	$n_{crh} \geq$	Hashlength \geq	Security
LLL 1.18	211	5500	2^{208}
LLL (average) 5.2	890	29500	
RSR 1.19	1950	67500	2^{261}
Primal-dual RSR 1.21	3000	78200	2^{340}

The reduction algorithms and the corresponding minimal values of the security parameters and the hash length are given in the following table:

The security of the Micciancio-Lyubashevsky's hash should be the security of hash function against the most powerful of the lattice reductions i.e. 2^{340} for $X^{3000} + \dots + X + 1$ against the primal-dual RSR 1.21. Unfortunately this algorithm makes some assumptions such as RA and GSA, and more importantly has a quite significant time complexity which makes that sometimes the reduction is not feasible in practice. That is why the results about the results about some other algorithms (see table 13) could be also of interest.

The assumptions: GSA is a worst-case property. Bases that do not satisfy GSA are easier to reduce [41] p.10. 2k-BKZ and primal-dual reductions perform much better under GSA heuristic than proven in worst case without heuristics see [41] p.3. The algorithm primal dual RSR is highly parallel and with polynomial time on average but not proven polynomial time [41].

6 EU-CMA2 Security of MSS (EU-CMA2 = existentially unforgeable under adaptive chosen message attack)

We will propose in this section one general expression of the MLMSS security against primal-dual RSR attacks and simplified expressions for the MLMSS security which make the following assumption:

Assumption 1

$$\mathcal{C}(f - SPP_{\mathcal{O}(n^4)}) \approx 2^n$$

6.1 Security estimation without assumptions of the Merkle Signature Scheme against an attacker who uses the primal-dual RSR reduction algorithm 1.21

From the theorem 1.22 (see Preliminaries) we have:

$$t_{Mss} = \max\{t_{crh} - (N + 1) \cdot t_H, t_{ots}\} - t_G - 2^N \cdot t_S$$

$$\epsilon_{Mss} = 2 \cdot \max\{\epsilon_{crh}, 2^N \cdot \epsilon_{ots}\}$$

and we should impose $t_{Mss} > 0$ and $\epsilon_{Mss} \leq 1$. For the following we will always use the variable n_{crh} when we speak about the security parameter of the hash function that we use for hashing the signature tree, and we will always use the variable n_{ots} when we consider the security parameter of the one-time signature scheme.

For the collision-resistance of the hash function we use the expression 4.5 from paragraph 4.5 we have:

$$\mathcal{C}(MLhash) = \min\left\{2^{25 + \frac{g^{-1}(n_{crh})}{4} \cdot \log_2\left(\frac{g^{-1}(n_{crh})}{11}\right)}, \frac{\mathcal{C}(f-SPP_{\gamma_{crh}})}{\left(\lceil \frac{n_{crh}}{2 \cdot k_{crh}} \cdot \log_2\left(\frac{k_{crh}}{6}\right) - \log_2(\gamma_{crh}) \rceil\right)}\right\}$$

and for the security of the one-time signature scheme 5.5 from paragraph 5.5 :

$$\mathcal{C}(MLots) = \min\left\{2^{27 + \frac{g^{-1}(n_{ots})}{4} \cdot \log_2\left(\frac{g^{-1}(n_{ots})}{11}\right)}, \frac{\mathcal{C}(f-SPP_{\gamma_{ots}})}{\left\lceil \frac{n_{ots}}{2 \cdot k_{ots}} \cdot \log_2\left(\frac{k_{ots}}{6}\right) - \log_2(\gamma_{ots}) \right\rceil}\right\}$$

Taking into account what we proved in the main theorem and the fact that

$$\mathcal{C}(f - SPP_{n^4}) \approx y(n_{crh}) \cdot \mathcal{C}(ML \text{ hash function}); y(n_{ots}) \cdot \mathcal{C}(ML \text{ one-time signature})$$

(see notations of the main theorem), we obtain the following practical proposition:

Proposition 6.1. (Practical proposition (without assumption 1) for estimating the security of the Merkle Signature Scheme against primal-dual RSR attacks) For $60 \geq N \geq 20$, $n_{crh} > 1840$, $n_{ots} > 3000$ the original Merkle signature is $(t_{Mss}, \epsilon_{Mss}, 2^N)$ existentially unforgeable under adaptive chosen message attacks using primal-dual RSR reduction, where t_{Mss} and ϵ_{Mss} are given by:

$$t = \max\left\{\frac{1}{1.465} \cdot \left(\mathcal{C}(ML \text{ hash function}) \cdot \left(\frac{y(n_{crh})-1}{y(n_{crh})}\right)\right)^{1/2} - (N+1) \cdot t_H, \frac{1}{1.465} \cdot \left(\mathcal{C}(ML \text{ one-time signature}) \cdot \left(\frac{y(n_{ots})-1}{y(n_{ots})}\right)\right)^{1/2}\right\} - t_G - 2^N \cdot t_S$$

$$\epsilon = 2 \cdot \max\left\{\left(\mathcal{C}(ML \text{ hash function}) / \left(2^5 \cdot \frac{y(n_{crh})}{y(n_{crh})-1}\right)\right)^{-1/2}, 2^N \cdot \left(\mathcal{C}(ML \text{ one-time signature}) / \left(2^5 \cdot \frac{y(n_{ots})}{y(n_{ots})-1}\right)\right)^{-1/2}\right\}$$

and $t > 0$, $\epsilon \leq 1$. Here $y(n) = (\lceil (n/5) - \log_2(n^4) \rceil)$ and where $\mathcal{C}(ML \text{ hash function})$ and $\mathcal{C}(ML \text{ one-time signature})$ are given as above (see 4.5, 5.5).

Remark. The estimation of the MLMSS security obtained using proposition 6.1 underestimates the security of MLMSS and can be improved for instance by varying the powers $\frac{1}{2}$ and $-\frac{1}{2}$ in the exponents in the previous expressions (remember that the only conditions of 1.22 are $t_{MLMSS} > 0$ and $\epsilon_{MLMSS} \leq 1$). We will see in the following paragraphs that proposition 6.1 under assumption 1 6 gives enough security (see table 14).

6.2 Security estimation (under assumption 1) of the Merkle Signature Scheme against an attacker who uses the primal-dual RSR reduction algorithm 1.21

From the theorem 1.22 (see Preliminaries) we have:

$$t_{Mss} = \max\{t_{crh} - (N + 1) \cdot t_H, t_{ots}\} - t_G - 2^N \cdot t_S$$

$$\epsilon_{Mss} = 2 \cdot \max\{\epsilon_{crh}, 2^N \cdot \epsilon_{ots}\}$$

and we should impose $t_{Mss} > 0$ and $\epsilon_{Mss} \leq 1$.

For the collision-resistance of the hash function we use the simplified expression 4.5 from paragraph 4.5 we have:

$$\mathcal{C}(MLhash) = 2^{(25 + \frac{g^{-1}(n_{crh})}{4} \cdot \log_2(\frac{g^{-1}(n_{crh})}{11}))}$$

and for the security of the one-time signature scheme 5.5 from paragraph 5.5 :

$$\mathcal{C}(MLots) = 2^{(27 + \frac{g^{-1}(n_{ots})}{4} \cdot \log_2(\frac{g^{-1}(n_{ots})}{11}))}$$

where $g(x) = x \cdot (3 + \log_2(x))$

Moreover for practical implementation of the Merkle Signature Scheme we use in practice $60 \geq N \geq 20$.

In order to satisfy the condition $\epsilon_{Mss} \leq 1$ we should have $\epsilon_{crh} < \frac{1}{2}$ and $\epsilon_{ots} < \frac{1}{2^{41}}$. Looking at paragraphs 4.4 and 5.4 (Table 7 and Table 12) we see that for practical values of the parameters n_{crh} and n_{ots} we have respectively $\mathcal{C}(MLhash) > 2^{240}$ and $\mathcal{C}(MLots) > 2^{340}$. Using the following theorem we could convert a $(2^s, \frac{1}{2^5})$ -secure problem into a $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ we will then have $\epsilon_{cr} < 2^{-120}$ and $\epsilon_{ots} < 2^{-170}$ and the condition $\epsilon_{Mss} = 2 \max\{\epsilon_{cr}, 2^N \cdot \epsilon_{ots}\} < 1$ will be satisfied.

Theorem 6.2. (Richard Lindner, Johannes Buchmann) If a problem is $(t, 1 - \epsilon)$ secure for some $\epsilon > 0$, then for any $1 > \delta > 0$ we can set $k = \lceil \frac{\log(\epsilon)}{\log(1-\delta)} \rceil$ and the same problem is $(t/k, \delta)$ secure.

Proof Δ We show the contrapositive. So assume in time t/k the probability of solving the problem is $Pr(\text{solving}) \geq \delta \sum_{i=0}^{k-1} (1-\delta)^i = 1 - (1-\delta)^k \geq 1 - \epsilon$

The last inequality is true since we picked $k \geq \frac{\log(\epsilon)}{\log(1-\delta)}$, which completes the proof. \square

In our case $\epsilon = \frac{1}{2^5}$, our problem is $(2^s, \frac{1}{2^5})$ -secure and we want to convert it to a problem which is $(-, 2^{-5-\frac{s}{2}})$ -secure. We should therefore choose $\delta = 2^{-5-\frac{s}{2}}$. We need to calculate $k = \lceil \frac{\log(\epsilon)}{\log(1-\delta)} \rceil$ and using the Taylor's expansion $\log(1-\delta) \approx -\delta$ for $\delta \ll 1$ we arrive at $k \approx \frac{1}{\delta}$ and therefore the $(2^s, \frac{1}{2^5})$ -problem is also $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ -secure.

Table 14: Security of the Merkle Signature Scheme against a primal-dual RSR attack 1.21 for different values of the security parameters n_{crh} (hash function), n_{ots} (one-time signature) and N (2^N number of signatures)

n_{crh}	$f_1(X)$	H-length ₁	n_{ots}	$f_2(X)$	H-length ₂	N	Mss Sec (\geq)
1847	$X^{1846} \dots + X + 1$	55816	3001	$X^{3000} \dots + X + 1$	78236	20	2^{289}
1861	$X^{1860} \dots + X + 1$	56277	3011	$X^{3010} \dots + X + 1$	78526	40	2^{291}
2048	$X^{2048} + 1$	57098	4096	$X^{4096} + 1$	104415	60	2^{370}

Analogously to the previous theorem on the (t, ϵ) -security we will prove a similar theorem for the non- (t, ϵ) -security (the contrapositive to the (t, ϵ) -security, for the definition see 1.5):

Theorem 6.3. *If a problem is not- $(t, 1 - \epsilon)$ -secure for some $\epsilon > 0$, then for any $1 > \delta > 0$ we can set $k = \lfloor \frac{\log(\epsilon)}{\log(1-\delta)} \rfloor$ and the same problem is not- $(t/k, \delta)$ secure.*

Proof Δ *We show the contrapositive. So assume in time t/k the probability of solving the problem is $\Pr(\text{solving}) \leq \delta \sum_{i=0}^{k-1} (1 - \delta)^i = 1 - (1 - \delta)^k \leq 1 - \epsilon$*

The last inequality is true since we picked $k \leq \frac{\log(\epsilon)}{\log(1-\delta)}$, which completes the proof. \square

Remark. In the table above we use different values for the parameters and we estimate the security. The values in the "Mss Security" column are an inferior limit to the security and can be improved with a wise choice of the quantities t_{ots} , t_{cr} , ϵ_{ots} , ϵ_{cr} (for instance through changing the powers $\frac{1}{2}$ and $-\frac{1}{2}$ with other powers in the main theorem).

6.3 Security estimation (under assumption 1) of the Merkle Signature Scheme against an attacker who uses the RSR reduction algorithm

From the theorem 1.22 (see Preliminaries) we have:

$$t_{Mss} = \max\{t_{\text{crh}} - (N + 1) \cdot t_H, t_{\text{ots}}\} - t_G - 2^N \cdot t_S$$

$$\epsilon_{Mss} = 2 \cdot \max\{\epsilon_{\text{crh}}, 2^N \cdot \epsilon_{\text{ots}}\}$$

and we should impose $t_{Mss} > 0$ and $\epsilon_{Mss} \leq 1$.

For the collision-resistance of the hash function we use expression 4.5 (paragraph 4.5):

$$\mathcal{C}(MLhash) = 2^{(30 + \frac{g^{-1}(n_{\text{crh}})}{4}) \cdot \log_2(\frac{g^{-1}(n_{\text{crh}})}{6})}$$

and for the unforgeability of the one-time signature scheme 5.5 (paragraph 5.5):

Table 15: Security of the Merkle Signature Scheme against a RSR (a BKZ type of attack) for different values of the security parameters n_{crh} (hash function), n_{ots} (one-time signature) and N (2^N number of signatures)

n_{crh}	$f_1(X)$	H-length ₁	n_{ots}	$f_2(X)$	H-length ₂	N	Security (\geq)
1201	$X^{1200} \dots + X + 1$	35070	1951	$X^{1950} \dots + X + 1$	69826	20	2^{209}
1213	$X^{1212} \dots + X + 1$	35456	1973	$X^{1972} \dots + X + 1$	70712	40	2^{212}
2048	$X^{2048} + 1$	57098	2048	$X^{2048} + 1$	67584	60	2^{313}

$$C(MLots) = 2^{(32 + \frac{g^{-1}(n_{ots})}{4} \cdot \log_2(\frac{g^{-1}(n_{ots})}{6}))}$$

where $g(x) = x \cdot (3 + \log_2(x))$

Moreover for practical implementation of the Merkle Signature Scheme we use in practice $60 \geq N \geq 20$.

In order to satisfy the condition $\epsilon_{Mss} \leq 1$ we should have $\epsilon_{crh} < \frac{1}{2}$ and $\epsilon_{ots} < \frac{1}{2^{41}}$. Looking at paragraphs 4.3 and 5.3 (Table 6 and Table 11) we see that for practical values of the parameters n_{crh} and n_{ots} we have respectively $C(MLhash) > 2^{161}$ and $C(MLots) > 2^{261}$. Using the following theorem we could convert a $(2^s, \frac{1}{2^5})$ -secure problem into a $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ we will then have $\epsilon_{crh} < 2^{-80}$ and $\epsilon_{ots} < 2^{-130}$ and the condition $\epsilon_{Mss} = 2 \max\{\epsilon_{crh}, 2^N \cdot \epsilon_{ots}\} < 1$ will be satisfied.

We now use the theorem 6.2 to convert our $(2^s, \frac{1}{2^5})$ -secure problem to a problem which is $(-, 2^{-5-\frac{s}{2}})$ -secure. We should therefore choose $\delta = 2^{-5-\frac{s}{2}}$. We need to calculate $k = \lceil \frac{\log(\epsilon)}{\log(1-\delta)} \rceil$ and using the Taylor's expansion $\log(1-\delta) \approx -\delta$ for $\delta \ll 1$ we arrive at $k \approx \frac{1}{\delta}$ and therefore the $(2^s, \frac{1}{2^5})$ -problem is also $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ -secure. The idea is the same as in the main theorem and the error that we commit with the approximation $\log(1-\delta) \approx -\delta$ is negligible.

Remark. See remark 6.2.

6.4 Security estimation (under assumption 1) of the Merkle Signature Scheme against an attacker who uses a LLL-type reduction algorithm

From the theorem 1.22 (see Preliminaries) we have:

$$t_{Mss} = \max\{t_{crh} - (N+1) \cdot t_H, t_{ots}\} - t_G - 2^N \cdot t_S$$

$$\epsilon_{Mss} = 2 \cdot \max\{\epsilon_{crh}, 2^N \cdot \epsilon_{ots}\}$$

and we should impose $t_{Mss} > 0$ and $\epsilon_{Mss} \leq 1$.

We have the following expressions for the collision resistance of the hash function and the security of the one-time signature scheme:

For the collision-resistance of the hash function we use the expressions from paragraph 4.1 :

Table 16: Security of the Merkle Signature Scheme against a LLL-type attack for different values of the security parameters n_{crh} (hash function), n_{ots} (one-time signature) and N (2^N number of signatures)

n_{crh}	$f_1(X)$	H-length ₁	n_{ots}	$f_2(X)$	H-length ₂	N	LLL Security (\geq)
128	$X^{128} + 1$	3752	211	$X^{210} + \dots + X + 1$	5520	20	2^{165}
137	$X^{136} + \dots + X + 1$	3160	223	$X^{222} + \dots + X + 1$	5887	20	2^{174}
256	$X^{256} + 1$	6040	256	$X^{256} + 1$	6144	60	2^{187}

$$\mathcal{C}(\text{MLhash}) = \frac{2^{n_{\text{crh}}} - n_{\text{crh}}^3 \cdot \log(n_{\text{crh}})}{\left(\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{crh}} - 1) - \log_2(8 \cdot \epsilon^2 \cdot n_{\text{crh}} \cdot \log^4(n_{\text{crh}})) \rceil\right)} - \log(n_{\text{crh}}) \cdot \left(\frac{2^{n_{\text{crh}}^3}}{3} + n_{\text{crh}}^2 (1 + \log(n_{\text{crh}}) \cdot \log \log(n_{\text{crh}})) + 2n_{\text{crh}}\right)$$

and for the unforgeability of the one-time signature scheme the expression from paragraph 5.1:

$$\mathcal{C}(\text{MLots}) = \frac{2^{n_{\text{ots}}} - n_{\text{ots}}^3 \log(n_{\text{ots}})}{\left(\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{ots}} - 1) - \log_2(80 \cdot \phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^3 / \lceil \log_2(n_{\text{ots}}) \rceil) \log_2^5(n_{\text{ots}}) \rceil\right)} - \log(n_{\text{ots}}) \cdot \left(\frac{2^{n_{\text{ots}}^3}}{3} + n_{\text{ots}}^2 (1 + \log(n_{\text{ots}}) \cdot \log \log(n_{\text{ots}})) + 2n_{\text{ots}}\right)$$

Moreover for practical implementation of the Merkle Signature Scheme we use in practice $60 \geq N \geq 20$.

In order to satisfy the condition $\epsilon_{Mss} \leq 1$ we should have $\epsilon_{\text{crh}} < \frac{1}{2}$ and $\epsilon_{\text{ots}} < \frac{1}{2^{41}}$. Looking at Tables 3 and 9 we obtain that for practical values of the parameters n_{crh} and n_{ots} we have respectively $\mathcal{C}(\text{MLhash}) > 2^{110}$ and $\mathcal{C}(\text{MLots}) > 2^{200}$. Using theorem 6.2 we could convert a $(2^s, \frac{1}{2^5})$ -secure problem into a $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ we will then have $\epsilon_{\text{crh}} < 2^{-55}$ and $\epsilon_{\text{ots}} < 2^{-100}$ and the condition $\epsilon_{Mss} = 2 \max\{\epsilon_{\text{crh}}, 2^N \cdot \epsilon_{\text{ots}}\} < 1$ will be satisfied.

In our case $\epsilon = \frac{1}{2^5}$, our problem is $(2^s, \frac{1}{2^5})$ -secure and we want to convert it to a problem which is $(-, 2^{-5-\frac{s}{2}})$ -secure. We should therefore choose $\delta = 2^{-5-\frac{s}{2}}$. We need to calculate $k = \lceil \frac{\log(\epsilon)}{\log(1-\delta)} \rceil$ and using the Taylor's expansion $\log(1-\delta) \approx -\delta$ for $\delta \ll 1$ we arrive at $k \approx \frac{1}{\delta}$ and therefore the $(2^s, \frac{1}{2^5})$ -problem is also $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ -secure. The idea is the same as in the main theorem and the error that we commit with the approximation $\log(1-\delta) \approx -\delta$ is negligible.

Remark. See remark 6.2.

6.5 Summary: Simplified Practical Propositions

6.5.1 A practical proposition for estimating the security of MLMSS against attacks with primal-dual RSR lattice reduction

The proposition that we used to calculate the values in table 14. The idea is to transform all of the $(2^s, \frac{1}{2^5})$ -problems into $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5-\frac{s}{2}})$ -secure problems.

Proposition 6.4. (Practical proposition for estimating the security of the Merkle Signature Scheme against primal-dual RSR attacks) For $60 \geq N \geq 20$, $n_{\text{crh}} > 1840$, $n_{\text{ots}} > 3000$ the original Merkle signature is $(t_{M_{SS}}, \epsilon_{M_{SS}}, 2^N)$ existentially unforgeable under adaptive chosen message attacks where $t_{M_{SS}}$ and $\epsilon_{M_{SS}}$ are given by:

$$t_{MLMSS} = \max\left\{\frac{1}{1.465} \cdot 2^{\frac{(25 + \frac{g^{-1}(n_{\text{crh}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{crh}})}{11}))}{2}}, \frac{1}{1.465} \cdot 2^{\frac{(27 + \frac{g^{-1}(n_{\text{ots}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{ots}})}{11}))}{2}}\right\} - 2^N \cdot t_S$$

and,

$$\epsilon_{MLMSS} = 2 \cdot \max\left\{2^{-5 - \frac{(25 + \frac{g^{-1}(n_{\text{crh}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{crh}})}{11}))}{2}}, 2^N \cdot 2^{-5 - \frac{(27 + \frac{g^{-1}(n_{\text{ots}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{ots}})}{11}))}{2}}\right\}$$

where $g(x) = x \cdot (3 + \log_2(x))$

Remark. It is important to remember the assumptions that we made in order to write the previous expressions which are summarized in the conclusions 4.5, 5.5. In particular we made the assumption 1 (see assumption 6) which simplifies the expression of the security.

If assumption 1 is not satisfied we should use the general expression of the security of the hash function and the one-time signature (see 4.5, 5.5 for the case of primal-dual RSR).

6.5.2 A practical proposition for estimating the security of MLMSS against attacks with RSR lattice reduction

The proposition that we used to calculate the values in table 15. The idea is to transform all of the $(2^s, \frac{1}{2^s})$ -problems into $(\frac{1}{1.465} \cdot 2^{\frac{s}{2}}, 2^{-5 - \frac{s}{2}})$ -secure problems.

Proposition 6.5. (Practical proposition for estimating the security of the Merkle Signature Scheme against RSR attacks) For $60 \geq N \geq 20$, $n_{\text{crh}} > 1200$, $n_{\text{ots}} > 1950$ the original Merkle signature is $(t_{M_{SS}}, \epsilon_{M_{SS}}, 2^N)$ existentially unforgeable under adaptive chosen message attacks where $t_{M_{SS}}$ and $\epsilon_{M_{SS}}$ are given by:

$$t_{MLMSS} = \max\left\{\frac{1}{1.465} \cdot 2^{\frac{(30 + \frac{g^{-1}(n_{\text{crh}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{crh}})}{6}))}{2}}, \frac{1}{1.465} \cdot 2^{\frac{(32 + \frac{g^{-1}(n_{\text{ots}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{ots}})}{6}))}{2}}\right\} - 2^N \cdot t_S$$

and for the unforgeability of the one-time signature scheme:

$$\epsilon_{MLMSS} = 2 \cdot \max\left\{2^{-5 - \frac{(30 + \frac{g^{-1}(n_{\text{crh}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{crh}})}{6}))}{2}}, 2^N \cdot 2^{-5 - \frac{(32 + \frac{g^{-1}(n_{\text{ots}})}{4} \cdot \log_2(\frac{g^{-1}(n_{\text{ots}})}{6}))}{2}}\right\}$$

where $g(x) = x \cdot (3 + \log_2(x))$

Remark. See the remark 6.5.1. If the assumption 1 is not satisfied use 4.5 and 5.5.

6.5.3 A practical proposition for estimating the security of MLMSS against attacker who uses only LLL-type lattice reductions, but not BKZ-type reduction

The proposition that we used to calculate the values in table 16. The idea is to transform all of the $(2^s, 1)$ -problems into $(2^{\frac{s}{2}}, 2^{-\frac{s}{2}})$ -secure problems.

Proposition 6.6. (Practical proposition for estimating the security of the Merkle Signature Scheme against LLL-attacks) For $60 \geq N \geq 20$, $n_{\text{crh}} > 110$, $n_{\text{ots}} > 200$ the original Merkle signature is $(t_{M_{SS}}, \epsilon_{M_{SS}}, 2^N)$ existentially unforgeable under adaptive chosen message attacks where $t_{M_{SS}}$ and $\epsilon_{M_{SS}}$ are given by:

$$t_{M_{SS}} = \max\left\{\frac{1}{1.465} \cdot \left(\frac{2^{n_{\text{crh}}} - n_{\text{crh}}^3 \cdot \log(n_{\text{crh}})}{\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{crh}} - 1) - \log_2(8 \cdot \epsilon^2 \cdot n_{\text{crh}} \cdot \log^4(n_{\text{crh}})) \rceil} - \log(n_{\text{crh}}) \cdot \left(\frac{2n_{\text{crh}}^3}{3} + n_{\text{crh}}^2(1 + \log(n_{\text{crh}}) \cdot \log \log(n_{\text{crh}})) + 2n_{\text{crh}})\right)^{\frac{1}{2}}, \frac{1}{1.465} \cdot \left(\frac{2^{n_{\text{ots}}} - n_{\text{ots}}^3 \log(n_{\text{ots}})}{\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{ots}} - 1) - \log_2(80 \cdot \phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{3/\lceil \log_2(n_{\text{ots}}) \rceil} \log_2^5(n_{\text{ots}})) \rceil} - \log(n_{\text{ots}}) \cdot \left(\frac{2n_{\text{ots}}^3}{3} + n_{\text{ots}}^2(1 + \log(n_{\text{ots}}) \cdot \log \log(n_{\text{ots}})) + 2n_{\text{ots}})\right)^{\frac{1}{2}}\right)\right\}$$

$$\epsilon_{M_{SS}} = 2 \cdot 2^{-5} \cdot \max\left\{\left(\frac{2^{n_{\text{crh}}} - n_{\text{crh}}^3 \cdot \log(n_{\text{crh}})}{\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{crh}} - 1) - \log_2(8 \cdot \epsilon^2 \cdot n_{\text{crh}} \cdot \log^4(n_{\text{crh}})) \rceil} - \log(n_{\text{crh}}) \cdot \left(\frac{2n_{\text{crh}}^3}{3} + n_{\text{crh}}^2(1 + \log(n_{\text{crh}}) \cdot \log \log(n_{\text{crh}})) + 2n_{\text{crh}})\right)^{-\frac{1}{2}}, 2^N \cdot \left(\frac{2^{n_{\text{ots}}} - n_{\text{ots}}^3 \log(n_{\text{ots}})}{\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{ots}} - 1) - \log_2(80 \cdot \phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{3/\lceil \log_2(n_{\text{ots}}) \rceil} \log_2^5(n_{\text{ots}})) \rceil} - \log(n_{\text{ots}}) \cdot \left(\frac{2n_{\text{ots}}^3}{3} + n_{\text{ots}}^2(1 + \log(n_{\text{ots}}) \cdot \log \log(n_{\text{ots}})) + 2n_{\text{ots}})\right)^{-\frac{1}{2}}\right)\right\}$$

which can fortunately be simplified for the ordinary values for n_{crh} and n_{ots} into:

$$t_{M_{SS}} = \max\left\{\frac{1}{1.465} \cdot \frac{2^{\frac{n_{\text{crh}}}{2}}}{\left(\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{crh}} - 1) - \log_2(8 \cdot \epsilon^2 \cdot n_{\text{crh}} \log^4(n_{\text{crh}})) \rceil\right)^{\frac{1}{2}}}, \frac{1}{1.465} \cdot \frac{2^{\frac{n_{\text{ots}}}{2}}}{\left(\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{ots}} - 1) - \log_2(80 \cdot \phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{3/\lceil \log_2(n_{\text{ots}}) \rceil} \log_2^5(n_{\text{ots}})) \rceil\right)^{\frac{1}{2}}}\right\}$$

and

$$\epsilon_{M_{SS}} = 2^{-4} \cdot \max\left\{\frac{2^{-\frac{n_{\text{crh}}}{2}}}{\left(\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{crh}} - 1) - \log_2(8 \cdot \epsilon^2 \cdot n_{\text{crh}} \log^4(n_{\text{crh}})) \rceil\right)^{-\frac{1}{2}}}, 2^N \cdot \frac{2^{-\frac{n_{\text{ots}}}{2}}}{\left(\lceil \frac{\log_2(\frac{4}{3})}{2} \cdot (n_{\text{ots}} - 1) - \log_2(80 \cdot \phi^3 \cdot n_{\text{ots}}^2 \cdot (\phi n_{\text{ots}})^{3/\lceil \log_2(n_{\text{ots}}) \rceil} \log_2^5(n_{\text{ots}})) \rceil\right)^{-\frac{1}{2}}}\right\}$$

6.6 Conclusion of this section

Some propositions for the Merkle - Micciancio - Lyubashevsky signature scheme are given in table 15 with the corresponding polynomials.

Table 16 represents the security of some polynomials against LLL -type attacks. The proposed polynomials are unfortunately not secure against RSR - type of attacks.

Also some practical formulas that one can use to estimate the security of a signature scheme for particular values of the security parameters n_{crh} and n_{ots} are proposed in paragraph 6.

Conclusion

The actual security of the Merkle-Micciancio-Lyubashevsky's signature scheme depends on the developments in the field of lattice-basis reduction algorithms. Moreover it is a well known fact that the LLL and BKZ algorithms in practice perform much better than the expected theoretical bound. Although this often helps cryptanalysts to knock unexpectedly public-key cryptosystems, it makes the life of designers of lattice-based cryptosystems much more difficult because the security parameters should be in practice larger than those calculated with the LLL and BKZ bounds.

As we see the values for the parameter of security n which ensures some security against lattice-based attacks is too large to be of practical usage ($n_{crh} \geq 2000$ and $n_{ots} \geq 3000$). We need other propositions and ideas in the area of lattice based cryptography which will ensure the security for smaller values of the security parameters.

7 Bibliography

References

- [1] *P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring*
- [2] *R. Merkle A certified Digital Signature pages 218-238. Springer-Verlag New York, Inc. New York, NY, USA, 1989.*
- [3] *J. Buchmann, C. Coronado, M. Döring, D. Engelbert, C. Ludwig, R. Overbeck, A. Schmidt, U. Vollmer, Ralph-Philip Weinmann Post-Quantum Signatures Technical Report 297, Cryptology ePrint Archive, 2004.*
<http://eprint.iacr.org/2004/297/>
- [4] *Vadim Lyubashevsky, Daniele Micciancio Generalized Compact Knapsacks are Collision Resistant Electronic Colloquium on Computational Complexity (ECCC) Report TR05-142, 2005.*
- [5] *Vadim Lyubashevsky, Daniele Micciancio Generalized Compact Knapsacks are Collision Resistant In Proceedings of ICALP, Part 2, volume 4052 of Lecture Notes in Computer Science, pages 144-155. Springer-Verlag, 2006.*
- [6] *D. Micciancio, Vadim Lyubashevsky Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. Foundations of Computer Science, 2002, Proceedings. The 43rd Annual IEEE Symposium on, pages 356-365, 2002.*
- [7] *D. Micciancio, Vadim Lyubashevsky Asymptotically Efficient Lattice-Based Digital Signatures*
- [8] *D. Micciancio Shortest Vector Problem (1982; Lenstra, Lenstra, Lovasz) www.cs.ucsd.edu/~daniele/papers/AlgoEncyclopedia.pdf*
- [9] *Arjen K. Lenstra, Hendrik W. Lenstra, Jr., Laszlo Lovász, Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515-534, 1982*
- [10] *R. Kumar, D. Sivakumar On polynomial approximations to the shortest lattice vector length*
- [11] *Oded Regev On the Complexity of Lattice Problems with Polynomial Approximation Factors*
<http://www.cs.tau.ac.il/~odedr/>

- [12] Richard Lindner *Current attacks on NTRU*
www.cdc.informatik.tu-darmstadt.de/~rlindner/publications/2006-03-ntru.pdf
- [13] W. Diffie, M. E. Hellman. *New directions in cryptography*
<http://www-ee.stanford.edu/~hellman/publications/24.pdf>
- [14] Miklosz Ajtai. *The shortest vector problem is NP-hard for randomized reductions. In Proceedings of the 30th annual ACM Symposium on Theory of Computing, pages 10-19. ACM Press, 1998.*
- [15] M. Ajtai, R. Kumar, D. Sivakumar, *A sieve algorithm for the shortest lattice vector problem. In Proceedings of the 33rd annual ACM Symposium on Theory of Computing, pages 601-610, ACM Press, 2001.*
- [16] Johannes Buchmann and Christoph Ludwig. *Practical lattice basis sampling reduction. Technical Report 072, Cryptology ePrint Archive, 2005.*
<http://eprint.iacr.org/2005/072/>.
- [17] LiDiA a C++ library for computational number theory.
<http://www.informatik.tu-darmstadt.de/TI/LiDiA/>.
- [18] Christoph Ludwig *Practical Lattice Basis Sampling Reduction. PhD thesis, Technische Universität Darmstadt, 2005.*
<http://elib.tu-darmstadt.de/diss/000640/>.
- [19] Alexander May and Joseph H. Silverman. *Dimension reduction methods for convolution modular lattices. In Silverman Cryptography and Lattices volume 2146 of Lecture Notes in Computer Science, Providence, RI, USA, 2001, Springer Verlag, pages 110-125*
- [20] Ravi Kannan. *Minkowski's convex body theorem and integer programming. Mathematics of Operations Research, 12(5):415-440, 1987.*
- [21] Ravi Kannan. *Improved algorithm for integer programming and related lattice problems. In Proceedings of the 15th annual ACM Symposium on Theory of Computing, pages 193-206. ACM Press, 1983.*
- [22] D. Aharonov and O. Regev. *Lattice problems in $NP \cap coNP$.*
- [23] Johannes Buchmann, Luis Carlos Coronado Garcia, Erik Dahmen, Martin Döring, and Elena Klintsevich *CMSS-An Improved Merkle Signature Scheme In Proceedings of INDOCRYPT 2006, 2006*
- [24] Luis Carlos Coronado Garcia *On the security and the efficiency of the Merkle signature scheme*

- [25] Luis Carlos Coronado Garcia *Provably Secure and Practical Signature Schemes* PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, 2005,
<http://elib.tu-darmstadt.de/diss/000642>
- [26] H.Koy. *Primale-duale segment-reduction*.
<http://www.mi.informatik.uni-frankfurt.de/research/papers.html> ,
2004
- [27] Peter van Emde Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Technical Report 04, Mathematische Institut, Iniversity of Amsterdam, 1981.
<http://staff.science.uva.nl/peter/vectors/abstract.html>.
- [28] Bettine Helfrich *Algorithms to construct Minkowski reduced and Hermite reduced bases*. *Theoretical Computer Science*, 41:125-139, 1985.
- [29] Alston S. Householder *Unitary triangularization of a nonsymmetric matrix*. *Journal of the ACM*, 5(4):339-342, 1958.
- [30] Henrik Koy and Claus Peter Schnorr. *Segment and strong segment LLL-reduction of lattice bases*.
<http://www.mi.informatik.uni-frankfurt.de/research/papers.html>,
2002.
- [31] Jeffrey C. Lagarias, Hendrik W. Lenstr, and Claus Peter Schnorr. *Korkine-Zolotarev bases and successive minima of a lattice and its reciprocal lattice*. *Combinatorica*, 10(4):333-348, 1990.
- [32] O.Regev *On the Complexity of Lattice Problems with Polynomial Approximation Factors - a survey paper prepared for the LLL+25 conference*.
- [33] Phong Q.Nguyen and Jacques Stern. *Lattice reduction in cryptography*
<http://www.di.ens.fr/pnguyen/pub.html>
- [34] Phong Q.Nguyen and Jacques Stern. *The two faces of lattices in cryptology*.
<http://www.di.ens.fr/pnguyen/pub.html>
- [35] Claus Peter Schnorr. *A hierarchy of polynomial time lattice basis reduction algorithms*. *Theoretical Computer Science*, 53:201-224, 1987.
- [36] Claus Peter Schnorr. *Block reduced lattice bases and successive minima*. *Combinatorics, Probability and Computing*, 4:1-16, 1994.

- [37] Claus Peter Schnorr. *Lattice reduction by random sampling and birthday methods.*
<http://www.mi.informatik.uni-frankfurt.de/research/papers.html>
- [38] Claus Peter Schnorr and M.Euchner. *Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming* 66:181-199, 1994.
- [39] Victor Shoup. *Number theory library (NTL) for C++*
<http://www.shoup.net/ntl/>.
- [40] Joseph H.Silverman. *Cryptography and Lattices, volume 2146 of Lecture Notes in Computer Science, Providence, RI, USA, 2001. Springer Verlag.*
- [41] Claus Peter Schorr. *Blockwise Lattice Basis Reduction Revisited 2006.*
<http://www.mi.informatik.uni-frankfurt.de/research/papers/Pridu1.pdf>
- [42] Phong Q. Nguyen, Damien Stehlé *LLL on the average 2006,*
<http://www.mi.informatik.uni-frankfurt.de/research/papers/Pridu1.pdf>
- [43] S.Goldwasser, S.Micali, R.Rivest *A digital signature scheme secure against adaptive chosen-message attacks*
- [44] D.Boneh, I.Mironov, V.Shoup *A secure signature scheme from bilinear maps Topics in Cryptography-CT-RSA, 2612:98-110, 2003*
- [45] S.Even, O.Goldreich, S.Micali *On-line/off-line digital signatures*
<http://www.ime.usp.br/rt/cranalysis/EvenGoldMicali.pdf>
- [46] J.A.Buchmann *Introduction to cryptography. Springer Verlag, 2004*
- [47] Shafi Goldwasser, Silvio Micali, Ronald L.Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks (Revision March 23, 1995)*
- [48] *PARI software for scientific computing.*
<http://pari.math.u-bordeaux.fr/download.html>
- [49] Shafi Goldwasser, Daniele Micciancio *Complexity of Lattice Problems: A Cryptographic Perspective The Kluwer International Series in Engineering and Computer Science, vol. 671.Kluwer Academic Publishers. March 2002.*
- [50] Golub, Gene H., and Van Loan, Charles F. *Matrix computations, 3rd edition, Johns Hopkins, Baltimore, 1996.*

[51] *A.M.Odlyzko The Rise and Fall of Knapsack Cryptosystems*
<http://www.dtc.umn.edu/~odlyzko/doc/arch/knapsack.survey.pdf>

A Appendix section 1

Use appendices only when you need them. For example to show off some code, or lengthy proof that didn't fit in the regular part of the thesis.

A.1 How to turn a collision resistant compression function into a collision resistant hash function

As the function proposed by Micciancio is not exactly a hash-function but a collision resistant compression function we need an algorithm for converting a collision resistant compression function into a collision resistant hash function. Such an algorithm is the Merkle transformation (see for instance [46] section 12.4 for more details).

A.2 Time complexity of the reduction used for $f - \text{IncSPP}_\gamma$ p.10 [4]

Lemma A.1. Step (2) p.10 [4] "generate a uniformly random coset of $I/\langle g \rangle$ and let v_i be a polynomial in that set" takes time $\mathcal{O}(n^2)$.

Proof Δ The group $I/\langle g \rangle$ is finite, but it is exponentially large, and so it is not completely trivial to generate a random element from it. Because the group is so large, we need somehow use its generators to generate the random element, and this is why lemma 2.1 [4] is necessary. Lemma 2.1 [4] takes time $\mathcal{O}(n^2)$ [49] in the security parameter, thus the result. \square

Lemma A.2. Step (3) p.10 [4] "generate $y_i \in \mathbb{R}^n$ such that y_i has distribution $\frac{\rho_s}{s^n}$ and consider y_i as a polynomial in $\mathbb{R}[x]$ " takes time $\mathcal{O}(n)$.

Proof Δ This step can not take more than $n \times$ the time necessary to generate a single element from the gaussian distribution which is some constant time that we note t_{gauss} . Therefore the overall time $t_3 = n \cdot t_{\text{gauss}}$ is a linear function of n . \square

Lemma A.3. Step (4) p.10 [4] "let w_i be the unique polynomial in $\mathbb{R}[x]$ of degree less than n with coefficients in the range $[0, p)$ such that $p \cdot (v_i + y_i) \equiv g \cdot w_i$ in $\mathbb{R}^n/\langle pg \rangle$ " takes time $\frac{2n^3}{3} + \mathcal{O}(n^2 \cdot \log(n) \cdot \log\log(n))$

Proof Δ The proof of this is in the proof of lemma 5.3 [4], and the algorithm for finding w_i is essentially gaussian elimination once you have the basis for the group $\langle pg \rangle$. The basis for the group can be gotten by doing polynomial multiplication. As we need around n polynomial multiplications to find the basis of the group $\langle pg \rangle$ and as every polynomial multiplication can be done in $\mathcal{O}(n \cdot \log n \cdot \log\log n)$ by using the fast Fourier Transform [4] p.2, the time for finding a basis for $\langle pg \rangle$ is $\mathcal{O}(n^2 \cdot \log n \cdot \log\log n)$. On the other hand the time needed for the gaussian elimination is $\frac{2n^3}{3}$ [50]. Thus the overall time is $\frac{2n^3}{3} + \mathcal{O}(n^2 \cdot \log(n) \cdot \log\log(n))$ \square

Lemma A.4. *Step (5) p.10 [4] ” $a_i = [w_i] \bmod p$ (where $[w_i]$ means round each coefficient of w_i to the nearest integer) ” takes time $\mathcal{O}(n)$.*

Proof \triangle *We should do at most n calculations of the rounding function each of which takes time t_{round} . So the overall time needed to perform step (5) is $t_5 = n \cdot t_{\text{round}}$. \square*

```
/* Some code for the hash function and the one-time signature */
```

```
package de.flexiprovider.pqc.mlms;

import java.io.IOException;

import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

public class AlmostRandomGenerator {

    public int generator() throws NumberFormatException, ParserConfigurationException,
    SAXException, IOException {
        // we take the value for N from the .xml file
        int N = Integer.parseInt(ConfigParser.getProperty("ValueofN"));
        /* values from D.Micciancio, Vadim Lyubashevsky Asymptotically Efficient Lattice-
        Based Digital Signatures  $s = \lceil (\log_2(N))^2 \rceil$  */
        int s = (int) Math.floor(Math.pow((Math.log(N)/Math.log(2)),2));

        double randnum = Math.random();
        System.out.println("s is: " +s);
        System.out.println("The random number is: " +randnum);
        int[] coeff;

        /*  $s_2 = 2^s$  */
        int s2 = (int) Math.pow(2,s);
        /* randnums2 is a random integer in the interval  $[0, 2^s]$  */
        int randnums2 = (int) Math.floor(randnum*s2);
        System.out.println("The scaled random number is: " +randnums2);
        int i = 1;
        while (Math.floor(randnums2/Math.pow(2,(s-i)))> 0){
            randnums2 = (int) (randnums2 - Math.pow(2,(s-i)));
            i++;
        }
        /* The algorithm returns a positive integer i such that i is chosen with probability  $2^{-i}$  */
        return i;
    }
}
```

```
package de.flexiprovider.pqc.mlms;

import java.io.IOException;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;

import javax.xml.parsers.ParserConfigurationException;
```

```
import org.xml.sax.SAXException;
```

```
public class ElementofR {
```

```
    /* returns an element h of R, where the set R is defined as in the article of  
    Micciancio and Lyubashevsky "Generalized Compact Knapsacks are Collision Resistant"  
    p.6 Section 5.1. */
```

```
    int[] getElement() throws NumberFormatException, ParserConfigurationException,  
    SAXException, IOException{
```

```
        int N = Integer.parseInt(ConfigParser.getProperty("ValueofN"));
```

```
        int d = (int) Math.ceil((Math.log(N)/Math.log(2)));
```

```
        int[] h = new int[N+1];
```

```
        for (int n = 1; n <= N; n++) {  
            h[n] = (int) ((-d) + Math.random() * 2 * d);  
        }
```

```
        return h;  
    }
```

```
}
```

```
package de.flexiprovider.pqc.mlms;
```

```
import java.io.IOException;
```

```
import javax.xml.parsers.ParserConfigurationException;
```

```
import org.xml.sax.SAXException;
```

```
public class GeneratorMLots {
```

```
    /* the set  $DK_{\{i\}}$  from the article from Micciancio and Lyubashevsky "Asymptotically efficient  
    lattice-based digital signatures p.9 */
```

```
    public int[][] generateDK(int i) throws NumberFormatException, ParserConfigurationException, SAXException,  
    IOException {
```

```
        int N = Integer.parseInt(ConfigParser.getProperty("ValueofN"));
```

```
        int p = (int) Math.pow((3*N),3);
```

```
        int m = (int) Math.ceil((Math.log(N)/Math.log(2)));
```

```
        int d = (int) (5*i*Math.pow(p,Math.pow(m, -1)));
```

```
        int [][] array = new int [m+1][N+1]; // ok
```

```

    ElementofR r = new ElementofR();

    for (int n = 1; n <= m-1; n++) {
        for (int j = 1; j <= N ; j++){
            array[n][j] = r.getElement()[j];
        }
    }

    System.out.println("One element of array "+ array[2][1]);
    System.out.println("One element of array "+ array[1][1]);
    return array;
}

```

```

public int[][] generateDL(int i) throws NumberFormatException, ParserConfigurationException,
    SAXException, IOException {
    /* the set DL_{i} from the article from Micciancio and Lyubashevsky "Asymptotically efficient
    lattice-based digital signatures p.9 */
    int N = Integer.parseInt(ConfigParser.getProperty("ValueofN"));
    int p = (int) Math.pow((3*N),3);
    int m = (int) Math.ceil((Math.log(N)/Math.log(2)));
    int d = (int) (5*i*N*3*Math.pow(p,Math.pow(m, -1)));

    int [][] array = new int [m+1][N+1]; // ok
}

```

```

    ElementofR r = new ElementofR();

    for (int n = 1; n <= m-1; n++) {
        for (int j = 1; j <= N ; j++){
            array[n][j] = r.getElement()[j];
        }
    }

    System.out.println("One element of array "+ array[2][1]);
    System.out.println("One element of array "+ array[1][1]);
    return array;
}
}

```

```

package de.flexiprovider.pqc.mlms;

import java.io.IOException;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;

import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

```

```

public class Hashfunction {
    /* implementation of the hash function from the article of Micciancio

```

and Lyubashevsky "Generalized Compact Knapsacks are Collision Resistant"

*/

```
int[][] getInstance() throws NumberFormatException, ParserConfigurationException, SAXException, IOException{
```

```
    int N = Integer.parseInt(ConfigParser.getProperty("ValueofN"));
```

```
    int p = (int) Math.pow(N,2);
```

```
    System.out.println("The value of p: "+ p);
```

```
    int[] polynomial = new int[N+1];
```

```
    if (ConfigParser.getProperty("polynomial").equals("X^{n}+1")){
```

```
        for (int n = 1; n <= (N-1); n++) {
```

```
            polynomial[n] = (int) (0);
```

```
        }
```

```
        polynomial[0] = (int) (1);
```

```
        polynomial[N] = (int) (1);
```

```
        for (int n = 0; n <= (N); n++) {
```

```
            System.out.print(" "+ polynomial[n]);
```

```
        }
```

```
    }
```

```
    else if (ConfigParser.getProperty("polynomial").equals("X^{n-1}+...+1")){
```

```
        for (int n = 0; n <= (N-1); n++) {
```

```
            polynomial[n] = (int) (1);
```

```
        }
```

```
        for (int n = 0; n <= (N-1); n++) {
```

```
            System.out.print(" "+ polynomial[n]);
```

```
        }
```

```
    }
```

```
    else {
```

```
        System.out.print("Not correct type in the XML file");
```

```
    }
```

```
    int m = (int) Math.ceil((Math.log(N)/Math.log(2)));
```

```
    System.out.println("m: " + m);
```

```
    int d = (int) Math.ceil((Math.log(N)/Math.log(2)));
```

```
    int [][] array = new int [m+1][N+1]; // ok
```

```
    ElementofR r = new ElementofR();
```

```
    for (int n = 1; n <= m-1; n++) {
```

```
        for (int j = 1; j <= N ; j++){
```

```
            array[n][j] = r.getElement()[j];
```

```
        }
```

```
    }
```

```
    return array;
}
}
```

```
package de.flexiprovider.pqc.mlms;
```

```
import java.util.Properties;
```

```
import javax.xml.parsers.ParserConfigurationException;
```

```
import java.io.IOException;
```

```
import org.xml.sax.SAXException;
```

```
/* A class written partly from Vangelis Karatsiolis */
```

```
public class ConfigParser
```

```
{
```

```
    /** The singleton reference. */
```

```
    private static ConfigParser self = null;
```

```
    private static XMLConfigurationParser parser = null;
```

```
    /**
```

```
     * Protected constructor for singleton purpose.
```

```
     */
```

```
    protected ConfigParser() {
```

```
    }
```

```
    /**
```

```
     * Returns the singleton instance of ConfigParser.
```

```
     * @return The singleton instance.
```

```
     */
```

```
    public static ConfigParser instance() {
```

```
        if (self == null) {
```

```
            self = new ConfigParser();
```

```
        }
```

```
        return self;
```

```
    }
```

```
    /**
```

```
     * Returns the string translated in the current language.
```

```
     * @param string The string to translate.
```

```
     * @return The translated string.
```

```
     */
```

```
    public static String getProperty(String key) throws ParserConfigurationException, SAXException, IOException {
```

```
        return instance().getParser().getAllProperties("tudcardkeyrecovery").getProperty(key);
```

```
    }
```

```
    public XMLConfigurationParser getParser ()throws ParserConfigurationException, SAXException, IOException {
```

```
        if (parser == null) {
```

```
            parser = new XMLConfigurationParser("MLMSignatureScheme.xml");
```

```
        }
```

```
        return parser;
```

```
}  
}
```

MLMSignatureScheme.xml

```
<MLMSignatureScheme>
```

```
<nonreduciblepolynomial>
```

```
  <param key="ValueofN" value="10"/>
```

```
  <param key="polynomial" value="X^{n-1}+...+1"/>
```

```
</nonreduciblepolynomial>
```

```
</MLMSignatureScheme>
```