Politecnico di Torino

Politecnico di Torino

Master's degree in Cybersecurity

Academic year 2024/2025

Graduation Session October 2025

Lyubashevsky's lattice-based digital signature schemes: analysis of the parameters and the rejection sampling technique

Supervisors:
Prof. Antonio J. DI SCALA
Dr. Andrea FLAMINI

Candidate:
Roberto PREVITALI

Abstract

Lattice-based cryptography has emerged as one of the most promising candidates for post-quantum secure systems, thanks to its solid hardness assumptions and efficiency compared to other approaches. A central challenge in this field has been the design of secure and practical digital signature schemes. Lyubashevsky's attempt to create such a scheme dates back in 2009 with the publication of his work "Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures", where he presents a digital signature scheme obtained through Fiat-Shamir Transform. This signature relies on collision-resistant hash functions and the idea of aborting steps of the algorithm that would produce data that would be included in the signature and would leak information about the secret key. This is crucial to guarantee the security of the signature scheme itself. Although conceptually innovative, the resulting scheme was not practical for deployment.

The turning point came with Lyubashevsky's paper "Lattice Signatures Without Trapdoors" (2012), where he presents a digital signature scheme based on the well-known Small Integer Solution (SIS) problem, and uses the rejection sampling technique to ensure that the signatures are statistically independent of the secret key, preventing any leakage. The abort technique introduced in "Fiat–Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures" can be viewed as an early form of rejection sampling, although it was only with "Lattice Signatures Without Trapdoors" that the method was explicitly formalized and in terms of gaussian rejection sampling.

The influence of this idea has been long-lasting. The lattice-based digital signature scheme Dilithium, selected in the NIST Post-Quantum Cryptography standardization process, is explicitly based on Lyubashevsky's rejection sampling framework. However, because gaussian sampling is difficult to implement both securely and efficiently, Dilithium adopts a simplified approach using only uniform distributions. This highlights both the power and the practical challenges of rejection sampling in lattice-based cryptography.

This thesis investigates Lyubashevsky's 2012 scheme in depth, with particular emphasis on the rejection sampling procedure: why it is required, how it guarantees independence of signatures from the secret key, and what trade-offs it introduces. A detailed analysis of the parameters is provided, establishing the relationship between those concerning the security assumptions and those concerning the efficiency of the protocol, in particular the rejection sampling step and the size of the signature.

Building on this foundation, the thesis also examines the "bimodal gaussian" idea, a refinement that further improves efficiency by modifying the sampling distribution. The impact of this approach is studied through parameter analysis and comparison with the original scheme, with additional consideration of the Ring-SIS assumption.

The results of this work clarify the role of rejection sampling and parameter selection in lattice-based signature design. By analyzing both the original and refined techniques, the thesis contributes to a better understanding of how these schemes balance security and efficiency, and how the ideas pioneered by Lyubashevsky continue to shape practical post-quantum signatures.

Acknowledgements

I wish to express my sincere gratitude to my supervisor, Prof. Antonio José Di Scala, for his guidance, constructive feedback, and constant support throughout the preparation of this thesis.

I am also thankful to my co-supervisor, Dr. Andrea Flamini, for his valuable suggestions and assistance, which have contributed to the development of this work.

Above all, I would like to thank my family for their encouragement and support throughout my studies and the writing of the thesis.

Table of Contents

List of Tables

List of Figures

Acronyms

CDF
Cumulative Density Function

LWE
Learning With Errors

PDF
Probability Density Function

PMF
Probability Mass Function

PPT
Probabilistic Polynomial Time

SIS
Small Integer Solution

std
standard deviation

Chapter 1

Introduction

In recent years, lattice-based cryptography has emerged as one of the most promising families of post-quantum cryptographic primitives. The growing consensus is that the currently deployed cryptographic systems — most notably RSA and elliptic curve cryptography — will no longer be secure in the presence of large-scale quantum computers, due to Shor's algorithm efficiently solving the underlying number-theoretic problems. For this reason, mathematicians and cryptographers looked for alternatives that rely on mathematical problems believed to be resistant to quantum attacks. Lattice problems, such as the Short Integer Solution (SIS) and the Learning with Errors (LWE) problems, are at the core of many proposals in this area. They combine strong worst-case to average-case hardness guarantees with efficient algorithms, making them attractive both in theory and in practice.

A particularly important goal of post-quantum cryptography is the design of digital signature schemes that are not only secure, but also efficient. Digital signatures are a pillar of modern security protocols, enabling authentication, integrity, and the base for non-repudiation. Achieving practical lattice-based signatures has proven challenging: Vadim Lyubashevsky introduced a novel approach — that sought to bypass trapdoor mechanisms — in his work "Fiat–Shamir with Aborts" (2009), where he proposed a digital signature scheme where certain transcripts were discarded — or "aborted" — with a certain probability designed to hide information about the secret key. This abort mechanism can be seen as an early form of rejection sampling, although it was not yet formalized in those terms.

This idea matured in "Lattice Signatures Without Trapdoors" (2012), where Lyubashevsky presented a signature scheme based on Gaussian rejection sampling. The key point is that rejection sampling can be used to enforce statistical independence between the signatures and the secret key, thus preventing leakage and ensuring security under the SIS assumption. This work was a turning point: it provided the theoretical foundation for lattice-based signatures that are compact, secure, and independent of trapdoor constructions.

The main objective of this thesis is to analyze and better understand the role of rejection sampling as well as the role of the parameters in lattice-based digital signatures, with particular attention to the scheme of Lyubashevsky. The work is divided into two main contributions. The first is a detailed study of Lyubashevsky's lattice-based signature scheme, covering its construction, the theoretical principles of rejection sampling, the security proof of the signature scheme, and an analysis of the parameters involved in the signature scheme, highlighting the delicate balance between security and efficiency.

Moreover, possible improvements are discussed when considering alternative hardness assumptions, such as low-density SIS or LWE.

The second contribution is the study of the bimodal Gaussian idea, which was introduced to further improve efficiency. This refinement, used in the BLISS (Bimodal Lattice Signature Scheme) scheme, modifies the sampling distribution, allowing for shorter signature sizes, moving in the direction of a more practical scheme. In this context, we analyze the parameters of the BLISS scheme, paying particular attention to how rejection sampling interacts with the new distribution and how this affects overall performance. Finally, we briefly comment on the empirical benchmarks presented in the BLISS paper, which compare its performance with classical signature schemes such as RSA and ECDSA. These comparisons provide insight into the practical competitiveness of lattice-based signatures against well-established standards.

In summary, this thesis provides an in-depth study of the rejection sampling technique in lattice-based digital signatures, both in its original Gaussian formulation and in its bimodal enhancement. By analyzing the parameters and efficiency trade-offs, the work sheds light on the design principles underlying practical post-quantum signatures and illustrates the lasting influence of Lyubashevsky's ideas on the development of secure and efficient cryptographic primitives.

1.1 Thesis structure

- Chapter 2: presents useful preliminaries about zero-knowledge proofs, identification schemes, and digital signatures. These preliminaries refer to the slides of the course "Advanced Cryptography" held in the academic year 2024-2025 by professor Carlo Sanna for the Cybersecurity's Master degree.
- Chapter 3: presents Lyubashevsky's digital signatures scheme with Gaussian rejection sampling, focusing on the rejection sampling technique and the analysis on the parameters. The chapter concludes with a discussion of how alternative assumptions such as low-density SIS and LWE can influence the scheme.
- Chapter 4: presents the bimodal Gaussian idea and the BLISS signature scheme, focusing on a new way of generating the keys and the analysis of the parameters. The chapter concludes with a brief comparison of the performance of BLISS against classical digital signature implementations such as RSA and ECDSA.
- Chapter 5: this chapter contains the final considerations about the rejection sampling technique and the digital signature scheme as well as a suggestion on how the thesis work could be extended investigating Lyubashevsky's techniques in lattice-based zero-knowledge proofs.
- Appendix A: contains some python scripts that were used to generate meaningful graphs shown in the chapters.
- Appendix B: contains an explanation of the problem for the possible extension of the thesis work.
- Appendix C: contains useful timestamps of online talks where Lyubashevsky tells about identification schemes, digital signatures, and zero-knowledge proofs.

Chapter 2

Preliminaries

2.1 Zero Knowledge Proofs

2.1.1 Interactive Proof (IP) Systems

Definition 2.1 (Interaction of functions)**.** *A k-round interaction of two functions $f, g : \{0,1\}^* \to \{0,1\}^*$ on input $x \in \{0,1\}^*$ is a sequence of strings $a_1, ..., a_k \in \{0,1\}*$ defined as*

$$a_1 := f(x)$$
$$a_2 := g(x, a_1)$$
$$a_3 := f(x, a_1, a_2)$$
$$\dots$$
$$a_{2i+1} := f(x, a_1, ..., a_{2i}) \quad \text{for } 2i < k$$
$$a_{2i+2} := g(x, a_1, ..., a_{2i+1}) \quad \text{for } 2i + 1 < k$$

The final output is of f is denoted by $out_V\langle f, g\rangle(x)$.

Definition 2.2 (IP Class)**.** *A language L is said to be in the complexity class IP if there exist a PPT algorithm V (verifier) that can have a k-round interaction with a function $P : \{0,1\}^* \to \{0,1\}^*$ (prover) such that the following two properties hold.*

- *Completeness. If $x \in L$, then there exists P such that $Pr[out_V\langle V, P\rangle(x) = 1] \geq 2/3$*
- *Soundness. If $x \notin L$, then for everyP we have that $Pr[out_V\langle V, P\rangle(x) = 1] \leq 1/3$*

A pair (V, P) satisfying completeness and soundness is called interactive proof system for L.

Informally, $L \in IP$ if, for every $x \in L$, there exists an interactive proof that, with overwhelming probability, convinces the verifier that $x \in L$; while, if x $\notin L$, such proof is impossible.

The class IP does not change if its definition is changed by:

- allowing the prover P to be probabilistic
- replacing the probability 2/3 with 1
- replacing the probability 1/3 with any fixed constant $c < 1$ (this follows by repeating the interactive proof several times). In this case, c is called soundness error or cheating probability.

2.1.2 Zero-Knowledge Proofs

Definition 2.3 (Zero-Knowledge Proof)**.** *Let L be a language in NP. A pair (P, V) of interactive PPT algorithms is a perfect (statistical, computational, resp.) zero-knowledge proof (ZKP) for L if the following three properties hold.*

- ***Completeness****. For every $x \in L$ and for every witness w of this fact, we have that $Pr[out_V\langle P(x,w), V(x)\rangle = 1] \geq 2/3$*
- ***Soundness****. For every $x \notin L$ and for every (unbounded) probabilistic algorithm $\tilde{P}$, we have that $Pr[out_V\langle \tilde{P}(x), V(x)\rangle = 1] \leq 1/3$*
- ***Zero-Knowledge****. For every PPT algorithm V^*, there exists an expected PPT algorithm S (simulator) such that for every $x \in L$ and for every witness w of this fact, the random variables $out_{V^*}\langle P(x,w), V^*(x)\rangle$ and $S(x)$ are identical (statistically indistinguishable, computationally indistinguishable, resp.).*

The completeness and soundness properties of ZKPs are analogous to that of IP and the same considerations apply. The zero-knowledge property says that the verifier cannot learn anything new from the interaction, even if he employs a different strategy V^*. Indeed, he could have obtained the same information by directly executing the simulator S on the publicy known input x.

Sometime the zero-knowledge property might be too hard to handle, since it takes into account every possible strategy of the verifier. Then, we can relax it to a weaker property, where the verifier is assumed to behave honestly.

- **Honest-Verifier Zero-Knowledge**. There exist an expected PPT algorithm S (simulator) such that for every $x \in L$ and for every witness w of this fact, the random variables $\text{out}_V\langle P(x,w), V^*(x)\rangle$ and $S(x)$ are identical (statistically indistinguishable, computationally indistinguishable, resp.).

The definition of zero-knowledge that we have seen can prove the existence of a witness w for a statement x in a NP-language L without revealing the witness w. However proving the existence of the witness w is much weaker than proving the knowledge of w.

Definition 2.4 (Proof of Knowledge)**.** *Let L be a language in NP. A pair (P, V) of interactive PPT algortihms is a proof of knowledge for L if the following propery holds:*

- ***Knowledge soundness*** *(or knowledge extractability). There exists a constant $k > 0$ (knowledge error) and an expected PPT algorithm E (extractor) such that for every interactive function $\tilde{P}$ and every $x \in \{0,1\}^*$ the following condition holds: if $p := Pr[out_V\langle \tilde{P}, V\rangle = 1] > k$ then, on input x and access to $\tilde{P(x)}$, the machine E returns a witness w for $x \in L$ within a number of steps bounded by $1/(p-k)$ times a fixed polynomial of $|x|$.*

Informally, the definition of Proof of knowledge says that given any algorithm $\tilde{P}$ that convinces the verifier that $x \in L$, it is possible to build another algorithm E that produces (extracts) a witness w for the fact that $x \in L$.

It can be proved that knowledge soundness implies soundness. At this point, zero knowledge proofs of knowledge (ZKPoK) can be defined as zero knowledge proofs which are also proofs of knowledge. Roughly speaking a honest prover is able to convince a verifier that he knows a witness w for a fact $x \in L$ (proof of knowledge property) and the verifier does not learn anything about the witness w (zero-knowledge property).

Definition 2.5 (Zero-Knowledge Proof of Knowledge)**.** *Let L be a language in NP. A pair (P, V) of interactive PPT algorithms is a perfect (statistical, computational, resp.) zero-knowledge proof of knowledge (ZKPoK) for L if the following three properties hold.*

- ***Completeness****. For every $x \in L$ and for every witness w of this fact, we have that $Pr[out_V\langle P(x,w), V(x)\rangle = 1] \geq 2/3$*
- ***Knowledge soundness****. There exists a constant $k > 0$ (knowledge error) and an expected probabilistic oracle machine E (extractor) such that for every interactive function $\tilde{P}$ and every $x \in L$ the following condition holds: If $p := Pr[out_V\langle \tilde{P}, V\rangle = 1] > k$ then, on input x and access to $\tilde{P}(x)$, the machine E returns a witness w for $x \in L$ within a number of steps bounded by $1/(p-k)$ times a fixed polynomial of $|x|$.*
- ***Zero-Knowledge****. For every PPT algorithm V^*, there exists an expected PPT algorithm S (simulator) such that for every $x \in L$ and for every witness w of this fact, the random variables $out_{V^*}\langle P(x,w), V^*(x)\rangle$ and $S(x)$ are identical (statistically indistinguishable, computationally indistinguishable, resp.).*

Informally,

- an honest prover that knows a valid witness w will convince the verifier with significant probability
- a cheating prover that is able to convince the verifier with significant probability and without knowing a valid witness w, can employ the extractor to recover a valid witness w(i.e. you can't cheat with significant probability without knowing a valid witness)
- the verifier learns nothing beyond the fact that the prover knows some witness w.

2.2 Σ-protocols and Identification schemes

2.2.1 Σ-protocols

A Σ-protocol is a particular Honest-Verifier ZKPoK with a 3-pass structure.

Definition 2.6 (Σ-protocol)**.** *Let L be a language in NP. A Σ-protocol is a 3-pass interactive proof between a prover and a verifier having the following structure:*

1. *The prover's input is a statement $x \in L$ and a witness w of that fact*
2. *The verifier's input is x*
3. *The prover sends to the verifier a **commitment***
4. *The verifier sends to the prover a **challenge** from a finite set of challenges $\mathcal{C}$*
5. *The prover computes an appropriate **response** and sends it to the verifier*
6. *The verifier checks if the response is correct, in according to the challenge and the commitment. In such a case the verifier accepts, otherwise the rejects.*

The prover consists of two PPT algorithms P_1, P_2, and the verifier consists of two PPT algorithms V_1, V_2. The scheme in figure 2.1 summarize the execution of the protocol of transcript (x, com, ch, rsp).

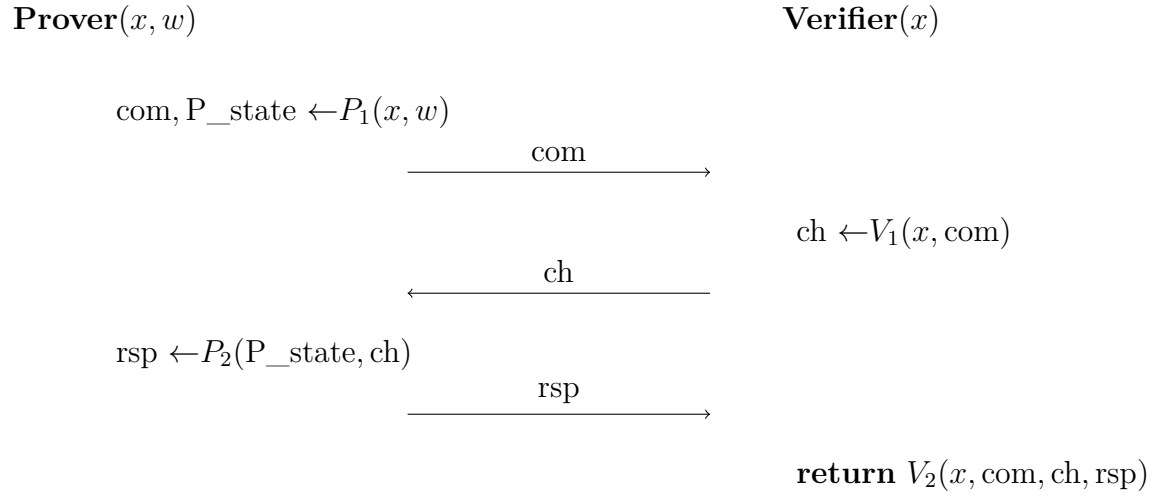


Figure 2.1: A generic Σ-protocol between a prover and a verifier.

Moreover, the following properties hold.

- **Completeness.** If the parties follow the protocol for $x \in L$ then the verifier always accepts.
- **Special Soundness.** There exists an expected PPT algorithm, called **extractor**, that takes as input two transcripts (x, com, ch_1, rsp_1) and (x, com, ch_2, rsp_2), with $\text{ch}_1 \neq \text{ch}_2$, of a honest execution of the protocol, and returns as output a witness w' of $x \in L$. (Here w and w' may not be equal).
- **(Perfect, Statistical, Computational) Honest Verifier Zero-Knowledge (HVZK).** There exists an expected PPT algorithm, called **simulator**, that takes as input x and returns as ouput a random transcript (x, com, ch, rsp) having identical (statistically indistinguishable, computationally indistinguishable, resp.) probability distribution of transcripts of a honest execution of the protocol on input (x, w).

Informally, the notion of special soundness expresses the idea that if a cheater is able to answer to more challenges for the same commitment, then he is also able to compute the witness thanks to the extractor. In other words, cheating is as difficult as computing the witness. The property of honest verifier zero-knowledge says that a simulator, who does not know the witness, can still produce transcripts that are indistinguishable from a legit execution of the protocol. Therefore, the transcripts of a legit execution of the protocol contain no information (zero-knowledge) on the witness.

As an example of Σ-protocol, we show the Schnorr Σ-protocol in figure 2.2. The prover constructs a cyclic subgroup G of $\mathbb{Z}_p^*$ of prime order q generated by g, generates a random $x \xleftarrow{\$} \mathbb{Z}_q$ and computes $h := g^x \pmod q$. The prover wants to convince the verifier that he knows $x \in \mathbb{Z}_q$ (witness) such that $h = g^x \pmod q$ (statement), but without revealing x.

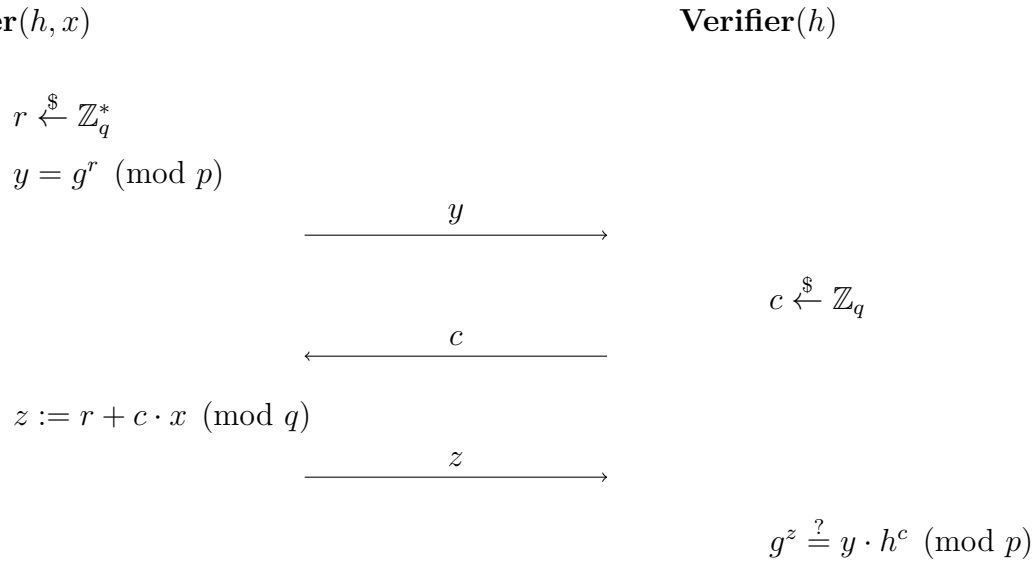


Figure 2.2: Schnorr Σ-protocol.

We can easily check that the Schnorr Σ-protocol satisfies the properties of completeness, special soundness and honest-verifier zero-knowledge.

Completeness. From the verification equation $g^z \stackrel{?}{=} y \cdot h^c$, if the prover and the verifier behaves honestly, then the verifier always accepts. Indeed, $g^z = g^{r+cx} = g^r \cdot g^{cx} = y \cdot (g^x)^c = y \cdot h^c$ since $y = g^r$ and $h = g^x$.

Special soundness. Given two valid transcripts of the same commitment (y, c_1, z_1) and (y, c_2, z_2) with $c_1 \neq c_2$, we are able to extract a witness, that is, an integer x' such that $h = g^{x'}$. Informally this means that if the prover is able to answer two different challenges for the same commitment, then he is also able to compute secret witness, that is, cheating is as difficult as computing the secret. Indeed, since the verifier accepts the transcripts, we have that $g^{z_1} = yh^{c_1}$ and $g^{z_2} = yh^{c_2}$. Dividing the first identity by the second one, we get $g^{z_1 - z_2} = h^{c_1 - c_2}$. Note that $c_1 \neq c_2$ implies that $c_1 - c_2$ is invertible in $\mathbb{Z}_q$. Therefore, if $(c_1 - c_2)^{-1}$ denotes the multiplicative inverse of $c_1 - c_2$ in $\mathbb{Z}_q$, then we get that $g^{(z_1 - z_2)(c_1 - c_2)^{-1}} = h$, and $x' = (z_1 - z_2)(c_1 - c_2)^{-1}$.

Honest verifier zero-knowledge. The interaction between prover and verifier does not leak any information about the secret to the verifier. Let us construct a Simulator $\mathbf{S}(h)$, which has access only to the public information h, that produces transcripts with the same distribution of legit transcripts. Note that in a legit execution of the protocol the transcript is (y, c, z), where y is a random element of a cyclic subgroup G of $\mathbb{Z}_p^*$ of prime order q; c is a random element of $\mathbb{Z}_q$; $z \in \mathbb{Z}_q$ satisfies $g^z = y \cdot h^c$. Equivalently, z is a random element of $\mathbb{Z}_q$; c is a random element of $\mathbb{Z}_q$; and $y \in G$ satisfies $g^z = y \cdot h^c$. Therefore, the simulator $\mathbf{S}(h)$ can be build as follows:

1. generate random $z \xleftarrow{\$} \mathbb{Z}_q$ and $c \xleftarrow{\$} \mathbb{Z}_q$
2. compute y $\leftarrow g^z \cdot h^{-c}$
3. return the transcript (y, c, z)

The transcripts generated in this way are legit (a posteriori, since they are constructed knowing the challenge from the beginning) and have the same probability distributions as the transcripts in a legit execution of the protocol.

2.2.2 Identification schemes

An identification scheme is an interactive method that allows a prover to prove its identity to a verifier. A very common way to construct an identification scheme is to use a Σ-protocol where the witness is a secret key and the statement is the public key associated with that secret key. The prover has the public key, which is publicly known, and the secret key, which he keeps for himself.

Definition 2.7 (Canonical Identification Scheme)**.** *A (3-pass) identification scheme consists of PPT algorithms Gen, P_1, P_2, V such that:*

- *The key-generation algorithm takes as input 1^λ, where λ is the security parameter, and returns as output the public key pk and the secret key of the prover*
- *The prover and the verifier runs P_1, P_2, V as in figure 2.3 and, if everybody behaves honestly, the verifier returns 1 (accept).*

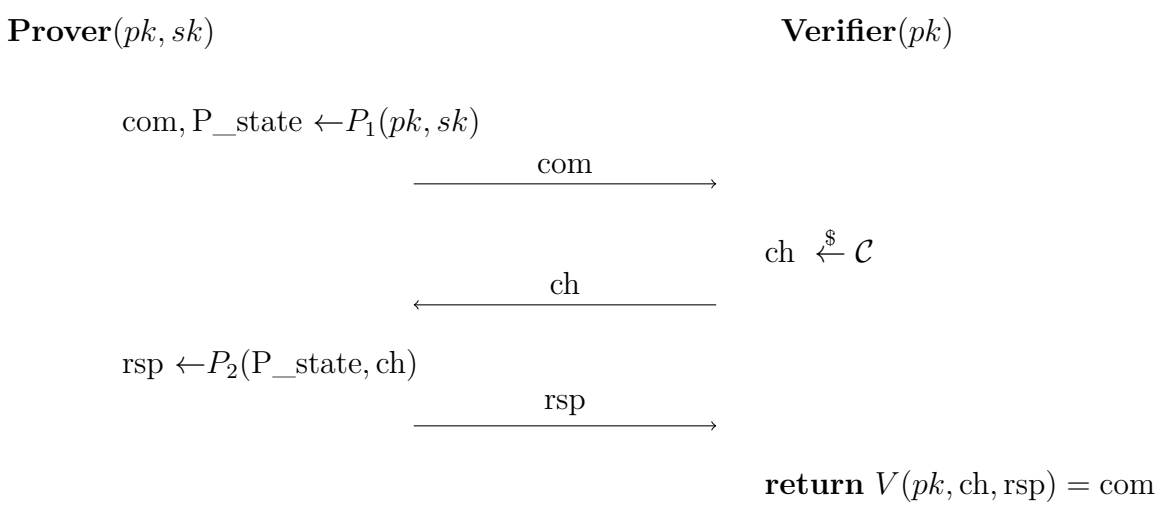


Figure 2.3: A generic identification scheme between a prover and a verifier.

The Schnorr Σ-protocol in Figure 2.2 can be regarded as an identification scheme as long as x is the secret key and h is the public key associated to it.

2.2.3 Secure identification schemes

Let $\Pi = (Gen, P_1, P_2, V)$ be an identification scheme. For every adversary $\mathcal{A}$, we define the experiment in figure 2.4.

Definition 2.8 (Secure Identification Scheme)**.** *An identification scheme Π is secure (against passive attacks) if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that $Pr[Ident_{\mathcal{A},\Pi}(\lambda) = 1] \leq negl(\lambda)$.*

As we mentioned above, a standard way to design canonical identification schemes is via $\Sigma-$protocol. The reason is that the properties of $\Sigma-$protocols guarantee that the derived identification scheme is secure against passive attacks, assuming that the language underlying the sigma protocol is hard (i.e. given a random statement $x \in L$ it is hard to find a witness w for that statement).

This can be proven by designing a reduction R that can break the hardness of the language as follows.

1. The prover keys are generated (pk,sk) $\leftarrow Gen(1^\lambda)$.
2. The adversary $\mathcal{A}$ is given as inputs 1^λ, where λ is the security parameter, and pk and access to an oracle $Trans_{sk}$ that when called runs an honest execution of Π and returns the transcript (com, ch, rsp).
3. The adversary $\mathcal{A}$ outputs a commitment com.
4. A random challenge ch $\leftarrow \mathcal{C}$ is generated and given to $\mathcal{A}$.
5. The adversary $\mathcal{A}$ outputs a response rsp.
6. The output of the experiment is 1 (success) if V(pk, ch, rsp) = 1, and 0 (failure) otherwise.

Figure 2.4: $\text{Ident}_{\mathcal{A},\Pi}(\lambda)$ experiment

Reduction overview. We assume there exist an adversary A for the experiment in Figure 2.4 that wins with non-negligible probability, then we show that our reduction R can use A to break the hardness of the language.

The reduction receives a random statement x in the language, and sets $pk = x$ and sends pk to A. Then, A can send to the transcript oracles queries. These queries will be answered by R who can simulate transcripts of the identification scheme since the sigma protocol is HVZK so there exists a simulator that produces transcripts indistinguishable from real executions of the Σ protocol and therefore of the identification scheme.

Eventually A sends to R a commitment com, and R sends back a random challenge c, and A with non-negligible probability produce a valid response z for that challenge. Then, the reduction R rewinds the adversary to the moment it has sent the commitment com and sends another challenge c'. According to the Forking lemma, if the challenge space is super-polynomial in size, A will produce a valid response z' with non-negligible probability.

At this point the reduction R knows 2 transcripts of the sigma protocol (com, c, z) and (com, c', z') for the same commitment, then thanks to the special soundness of the sigma protocol it can compute a valid witness w' for the statement x.

2.3 Digital Signatures

Digital signatures provide a secure method for verifying the authenticity and integrity of digital documents. They are required to satisfy three fundamental properties:

- **Authenticity**, which ensures that the message have been signed by the claimed source.
- **Integrity**, which gauarantees that the message has not been altered or tampered during transmission. Any modification to the data after signing will invalidate the signature.
- **Non-repudiation**, which ensures that, once a message has been signed, the signer cannot later claim that he did not sign it.

Definition 2.9 (Digital Signature)**.** *A digital signature is a triple(Gen, Sign, Verify) of PPT algorithms such that:*

- *The key-generation algorithm Gen takes as input 1^λ, where λ is the security parameter, and outputs a pair of keys (pk,sk), where pk is the public key and sk is the secret key.*
- *The signing algorithm Sign takes as input a secret key sk and a message m (in the message space), and outputs a signature σ (in the signature space). Denote this operation as $\sigma \leftarrow Sign_{sk}(m)$.*
- *The verification algorithm Verify, which is deterministic, takes as input a public key pk, a message σ and outputs a bit b. Denote this operation as $b \leftarrow Verify_{pk}(m, \sigma)$.*

2.3.1 Secure digital signature

Let $\Pi = (Gen, Sign, Verify)$ be a digital signature. For every adversary $\mathcal{A}$, we define the **signature forging experiment** Sig-Forg$_{\mathcal{A},\Pi}(\lambda)$ in figure 2.5.

1. A pair of keys is generated (pk,sk) $\leftarrow Gen(1^\lambda)$.
2. The adversary $\mathcal{A}$ is given as inputs pk and access to an oracle $Sign_{sk}(\cdot)$.
3. The adversary $\mathcal{A}$ outputs a message-signature pair (m, σ).
4. The output of the experiment is 1 (success) if $Verify_{pk}(m, \sigma) = 1$ and $\mathcal{A}$ never queried m to the oracle $Sign_{sk}(\cdot)$, and 0 (failure) otherwise.

Figure 2.5: Sig-Forg$_{\mathcal{A},\Pi}(\lambda)$ experiment

Definition 2.10 (Secure Digital Signature)**.** *A digital signature Π is existentially unforgeable under an adaptive chosen-message attack, or secure, if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that $Pr[Sig - Forg_{\mathcal{A},\Pi}(\lambda) = 1] \leq negl(\lambda)$*

Hash-and-Sign paradigm

For efficiency reasons, in practice it is more convenient to sign a hash H(m) of the message m instead of the message itself. When using the hash-and-sign paradigm, to achieve a secure signature scheme, the hash function H must be a collision-resistant hash function.

2.3.2 Fiat-Shamir Transform

Let $\Pi = (Gen, P_1, P_2, V)$ be an identification scheme, and let H be a hash function whose outputs are challenges of Π. The Fiat-Shamir transform of Π is a signature scheme $(Gen, Sign, Verify)$ defined as follows.

- $Sign_{sk}(m)$
  1. compute the commitment $(com, state) \leftarrow P_1(pk, sk)$
  2. compute the challenge $ch \leftarrow H(com, m)$
  3. compute the response $rsp \leftarrow P_2(state, ch)$

4. return $\sigma := (ch, rsp)$

- $Verify_{pk}(m, \sigma)$
  1. compute the commitment $com \leftarrow V(pk, ch, rsp)$
  2. return 1 if $H(com, m) = ch$, and 0 otherwise.

The Schnorr digital signature in Figure 2.6 is obtained by applying the Fiat-Shamir transform to the Schnorr identification scheme. The secret key is $x \in \mathbb{Z}_p$ and the public key is $h := g^x \pmod q$.

- $Sign_{sk}(m)$
  1. generate a random $s \in \mathbb{Z}_p$ and compute the commitment $k \leftarrow g^s$
  2. compute the challenge $c \leftarrow H(k, m) \in \mathbb{Z}_p$
  3. compute the response $z \leftarrow s + c \cdot x$
  4. return the signature $\sigma := (c, z)$
- $Verify_{pk}(m, \sigma)$
  1. compute the commitment $k \leftarrow g^r h^{-c}$
  2. return 1 if $H(k, m) = c$, and 0 otherwise.

Figure 2.6: Schnorr's digital signature obtained via Fiat-Shamir

For the notion of "Security of the Fiat-Shamir Transform" as in Thereom 2.15, we need to first define what Random Oracles are.

Definition 2.11 (Random Oracle)**.** *A random oracle is an oracle that takes as input an arbitrary string $x \in \{0,1\}^*$ and returns as outputs a fixed-length string $y \in \{0,1\}^l$ according to the following rules.*

- *if x has not been queried before to the oracle, then the output y is taken at random from $\{0,1\}^l$ with uniform distribution, independently of any previous query, and (x,y) is recorded.*
- *if x has been queried before to the oracle, then the output y is the same as the previous queries with input x*

Lemma 2.12. *If a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ is modeled as a random oracle, then H is one-way.*

Lemma 2.13. *If a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ is modeled as a random oracle, then H is collision-resistant.*

Lemma 2.14. *Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be modeled as a random oracle, and let $x \in \{0,1\}^*$. An adversary $\mathcal{A}$ can learn H(x) with a non-negligible probability only if $\mathcal{A}$ queries x to the random oracle H (or $\mathcal{A}$ interacts with an entity querying x to the random oracle H).*

Theorem 2.15 (Security of the Fiat-Shamir Transform)**.** *Let Π be an identification scheme and let Π' be the digital signature obtained by applying the Fiat-Shamir Transform to Π with the hash function H. If Π is secure and H is modeled as a random oracle, then Π' is secure.*

2.4 Statistical distance

The following definitions are taken from [1, Section 3.11].

Definition 2.16 (Statistical distance)**.** *Suppose P_0 and P_1 are probability distributions on a finite set $\mathcal{R}$. Then their statistical distance is defined as*

$$\Delta[P_0, P_1] := \frac{1}{2} \sum_{r \in \mathcal{R}} |P_0(r) - P_1(r)|.$$

Definition 2.17 (Statistical indistinguishability)**.** *Let P_0 and P_1 be probability distributions on a finite set $\mathcal{R}$. We say that P_0 and P_1 are statistically indistinguishable (or statistically close) if the statistical distance $\Delta[P_0, P_1]$ is negligible.*

Chapter 3

Lattice-based digital signature

In this chapter we will analyze the lattice-based digital signature developed by Lyubashevsky in "Lattice Signatures Without Trapdoor" [2]. The main goal of Lyubashevsky was to create an efficient digital signature scheme obtained through Fiat-Shamir Transform rather than by means of lattice trapdoors, like in the *hash and sign* pagadigm as in the GPV digital signature scheme [3], that was quite inefficient.

Lyubashevsky uses as starting point for the digital signature scheme the results of its work "Fiat Shamir With Aborts" [4] in which he develops a identification scheme and the relative digital signature obtained through Fiat Shamir Transform based on the hardness of lattice problems. The negative aspect of that signature scheme is that the signature lengths are on the order of 50.000 - 60.000 bits, therefore the scheme was quite far from being practical.

The digital signature scheme in [2] has been also discussed by Lyubashevsky in some talks -for which it's possible to view the video recording-, in particular the talks from 2013 [5] and 2016 [6] helped us to understand the digital signature scheme, and for this reason useful (hopefully) timestamps about the video will be presented in the Appendix C.

This chapter is structured as follows. Section 1 present the signature scheme and the security assumptions on which it relies on; section 2 and 3 explain what is the rejection sampling techniques, how it works and why it works; section 4 describes the gaussian rejection sampling used in the signature scheme, along with the proof of security that is strictly related to the rejection sampling; section 5 is dedicated to the analysis of the parameters involved in the signature scheme; section 6 present a variant of the signature scheme based on another security assumption.

3.1 The signature scheme

Before presenting the signature scheme let us provide some definitions about the Normal distribution.

Definition 3.1. *The continuous Normal distribution over $\mathbb{R}^m$ centered at $\mathbf{v}$ with standard deviation σ is defined by the function $\rho^m_{\mathbf{v},\sigma}(\mathbf{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m e^{\frac{-||\mathbf{x}-\mathbf{v}||^2}{2\sigma^2}}$.*

When $\mathbf{v} = 0$ we write $\rho^m_\sigma(\mathbf{x})$. The discrete Normal distribution over $\mathbb{Z}^m$ is defined as follows:

Definition 3.2. *The discrete Normal distribution over $\mathbb{R}^m$ centered at some $\mathbf{v} \in \mathbb{Z}^m$ with standard deviation σ is defined as $D^m_{\mathbf{v},\sigma}(\mathbf{x}) = \rho^m_{\mathbf{v},\sigma}(\mathbf{x})/\rho^m_\sigma(\mathbb{Z}^m)$.*

The quantity $\rho_\sigma^m(\mathbb{Z}^m) = \sum_{\mathbf{z}\in\mathbb{Z}^m} \rho_\sigma^m(\mathbf{z})$ is a scaling quantity needed to make the function into a probability distribution.

The signature scheme from [2] is reported in Figure 3.1. The signer holds as secret key a matrix $\mathbf{S} \in \mathbb{Z}^{m\times k}$ of random integers with maximum absolute value d and the public key consists of a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and another matrix $\mathbf{T} \in Z_q^{n\times k}$. Note that the elements in $\mathbb{Z}_q$ are represented by integers in the range $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$. The matrix $\mathbf{A}$ can be shared among all users, but the matrix $\mathbf{T}$ is individual because it depends on the specific secret $\mathbf{S}$ different for each individual. The hash function H outputs a vector of k short coefficients, but with the limitation that only κ of them are non-zero elements. The security level of H is represented by k.

Signing Key: $\mathbf{S} \xleftarrow{\$} \{-d, \ldots, 0, \ldots, d\}^{m\times k}$
Verification Key: $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \quad \mathbf{T} \leftarrow \mathbf{AS} \pmod q \in \mathbb{Z}_q^{n\times k}$
Random Oracle: $H : \{0,1\}^* \to \{\, \mathbf{v} \in \{-1,0,1\}^k : \|\mathbf{v}\|_1 \le \kappa \,\}$

Sign$(\mu, \mathbf{A}, \mathbf{S})$

1: $\mathbf{y} \xleftarrow{\$} D_\sigma^m$
2: $\mathbf{c} \leftarrow H(\mathbf{Ay}, \mu)$
3: $\mathbf{z} \leftarrow \mathbf{Sc} + \mathbf{y}$
4: output $(\mathbf{z}, \mathbf{c})$ with probability $\min\left(\frac{D_\sigma^m(\mathbf{z})}{M \cdot D_{\mathbf{Sc},\sigma}^m(\mathbf{z})}, 1\right)$

Verify$(\mu, \mathbf{z}, \mathbf{c}, \mathbf{A}, \mathbf{T})$

1: Accept iff $\|\mathbf{z}\| \le \eta\sigma\sqrt{m}$ and $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc}, \mu)$

Figure 3.1: Signature Scheme

To sign a message μ the signer randomly picks an m-dimensional vector $\mathbf{y}$ from the distribution D_σ^m, for some standard deviation σ, then computes the challenge $\mathbf{c} = H(\mathbf{Ay}, \mu)$, and finally computes the response $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$.

The output will be the signature pair $(\mathbf{z}, \mathbf{c})$, but it's important that the distribution of $(\mathbf{z}, \mathbf{c})$ should be independent of the secret key $\mathbf{S}$, therefore rejection sampling is employed in step (4) of the signing algorithm to decide whether a signature pair should be given in output or rejected. The main goal of using the rejection sampling technique is to make $\mathbf{z} = \mathbf{Sc} + \mathbf{y} \sim D_{\mathbf{Sc},\sigma}^m$ distributed according to D_σ^m, independently of the secret key $\mathbf{S}$. This is crucial to prove the security of the signature scheme.

The general idea of the rejection sampling technique will be explained in Section 3.2, while more details about the rejection sampling with discrete Normal distribution - the technique used in the digital signature scheme - will be discussed in Section 3.4.

For the verification algorithm, we check that $||\mathbf{z}|| < \eta\sigma\sqrt{m}$, and since $\mathbf{Ay} = \mathbf{Az} - \mathbf{Tc}$, the signature will be accepted.

3.1.1 The SIS problem and its variants

The security assumptions upon which the signature scheme is based fall into the category of Small Integer Solution (SIS) problems defined in [2, Section 3].

Definition 3.3 (ℓ_2-SIS$_{q,n,m,\beta}$ **problem**). *Given a random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ *find a vector* $\mathbf{v} \in \mathbb{Z}^m \backslash \{0\}$ *such that* $\mathbf{Av} = 0$ *and* $||\mathbf{v}|| \le \beta$.

To guarantee that a solution to the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem exists, we require that $\beta \geq \sqrt{m}q^{n/m}$.

In Section 3.6 we will base the security of the signature scheme on the variants of the SIS problem defined below, which in turn will allow us to obtain shorter signatures.

Definition 3.4 ($\text{SIS}_{q,n,m,d}$ **distribution**). *Choose a random matrix* $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ *and a vector* $\mathbf{s} \xleftarrow{\$} \{-d, ...,0, ..., d\}^m$ *and output* $(\mathbf{A}, \mathbf{As})$*.*

Definition 3.5 ($\text{SIS}_{q,n,m,d}$ **search problem**). *Given a pair* $(\mathbf{A}, \mathbf{t})$ *from the* $SIS_{q,n,m,d}$ *distribution, find a* $\mathbf{s} \in \{-d, ...,0, ...d\}^m$ *such that* $\mathbf{As} = \mathbf{t}$*.*

Definition 3.6 ($\text{SIS}_{q,n,m,d}$ **decision problem**). *Given a pair* $(\mathbf{A}, \mathbf{t})$ *decide, with non-negligible advantage, whether it came from the the* $SIS_{q,n,m,d}$ *distribution or whether it was generated uniformly at random from* $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$*.*

The $\text{SIS}_{q,n,m,d}$ search (and decision) problems look different depending on the relationship between its parameters. If $d \ll q^{n/m}$, then there is a high probability that there exist only one vector $\mathbf{s}$ whose coefficients have maximum absolute value d such that $\mathbf{As} = \mathbf{t}$, and such instances of the $\text{SIS}_{q,n,m,d}$ problems are called *low-density* instances. On the contrary if $d \gg q^{n/m}$, then there are many possible vectors $\mathbf{s}$ such that $\mathbf{As} = \mathbf{t}$ because the $\text{SIS}_{q,n,m,d}$ distribution is statistically close to uniform over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$ (by the leftover hash lemma). In this case the instances of the $\text{SIS}_{q,n,m,d}$ problem are called *high-density* instances.

3.2 Rejection Sampling

Rejection sampling is a basic technique that comes from numerical analysis and computational statistics also knows as "reject-accept" method[1]. In this section we will explain the general concepts [8][9] and a simple application of this technique.

3.2.1 What is rejection sampling?

Assume we wish to generate samples from a **target** distribution $f(x)$, but only have access to a **proposal** distribution $g(x)$. The idea of rejection sampling is to use samples from g and filter them so that the accepted ones statistically follow f, as if they were drawn directly from f.

This means that once we get a sample from $g(x)$ we need to have a criteria to ensure the previously described property. Let's continue the discussion with an example.

Suppose that our target $f(x)$ is the Beta(α,β)-distribution on the interval [0,1]. The Beta(α,β)-distribution PDF is

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)},\ 0 \leq x \leq 1,$$

where

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}.$$[2]

[1] John von Neumann was already talking about the reject-accept methods in 1947. [7]

The mode of a Beta(α,β)-distribution, that is the point where the PDF reaches its maximum, is given by the formula $\left(\frac{\alpha-1}{\alpha+\beta-2}\right)$ for $\alpha > 1, \beta > 1$.

Thus, for the distribution $Beta(2,4)$ in Figure 3.2 we have that $f(x) = 20x(1-x)^3$ for $0 \leq x \leq 1$, and the mode is 0.25.

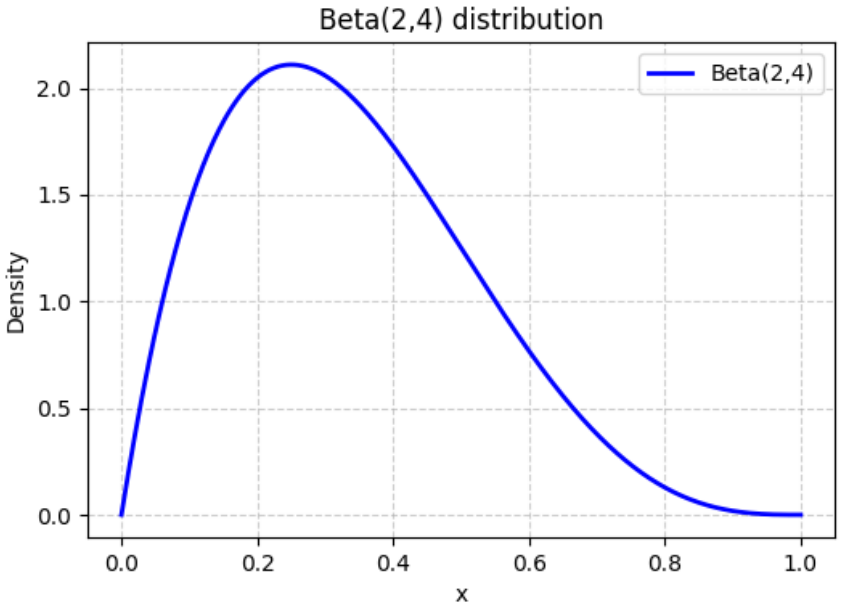


Figure 3.2: PDF of the Beta(2,4)-distribution

Now, as per the proposal distribution $g(x)$, we choose the uniform distribution on [0,1] -to match the domain of of our $Beta(2,4)$ distribution- in Figure 3.3. In general, we can use as proposal distribution any distribution we have access to.

At this point, a very important step to perform rejection sampling is to scale the proposal distribution $g(x)$ so that it encapsulates the target distribution $f(x)$. For this reason, we introduce a scaling constant M such that $M \geq \frac{f(x)}{g(x)}$, $\forall x \in [0,1]$.

The scaling constant ensures that the $0 < f(x)/(Mg(x)) \leq 1$, but we don't want M to be too high because, as we will see, it determines the expected number of iterations of the rejection sampling algorithm. That's why usually M is chosen as the upper limit of $f(x)/g(x)$.

In our example it's easy to see that we just have to scale the $Uniform(0,1)$ distribution just right above the maximum value of the $Beta(2,4)$ distribution in correspondence of its mode.

In particular we need

$$M \geq \frac{f(x)}{g(x)} = \frac{20x(1-x)^3}{1}, \forall x \in [0,1].$$

Thus, $M = 2.2 \geq max(f(x)) \approx 2.1093$ will do. In Figure 3.4 we can see the plot of both the $Beta(2,4)$ distribution and the scaled $Uniform(0,1)$ distribution.

[2]For a positive integer n, $\Gamma(n) = (n-1)!$

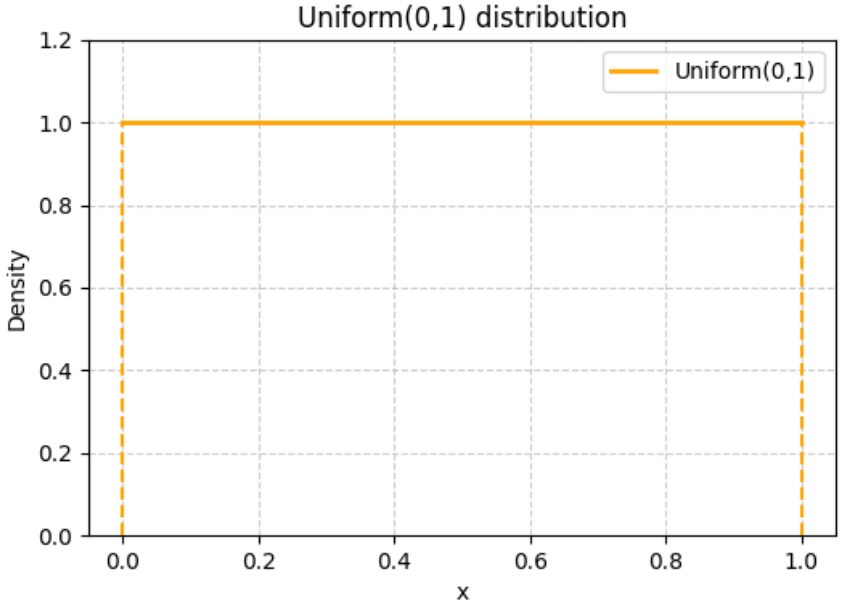


Figure 3.3: PDF of the Uniform(0,1)-distribution

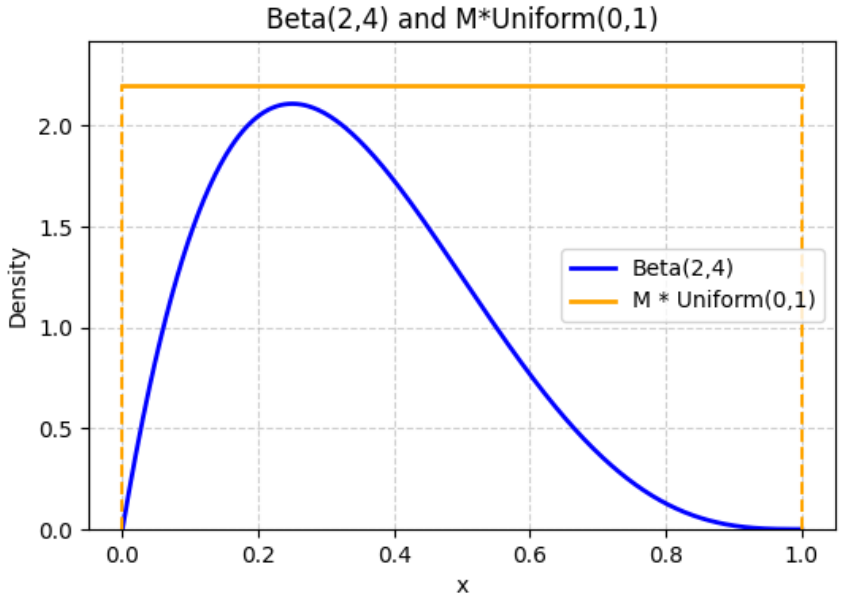


Figure 3.4: PDFs of the Beta(2,4) and scaled Uniform(0,1) distributions

3.2.2 How does rejection sampling work?

The steps of rejection sampling are typically presented in the following way:

1. sample x from the distribution $g(x)$
2. sample u from the distribution $Uniform(0, Mg(x))$
3. accept x if $u \leq f(x)$

In this way, the accepted sample will look like coming from the target distribution

$f(x)$, even if we didn't directly sample from it. Now we will explain these steps and in the section 3.2.3 we will give formal proofs of the correctness of the approach.

Let's have a look at the plot in Figure 3.5 where we sampled $x_0 = 0.5$. Should we accept this sample or not? We know that $g(x_0)$ tells us how likely is that our samples land near x_0. Intuitively, we are more likely to accept x_0 if $f(x_0) \approx Mg(x_0)$ because the likelihood of drawing a sample from the distribution $f(x)$ would be approximately the same, otherwise we are more likely to reject x_0 the more $Mg(x_0)$ differs from $f(x_0)$.

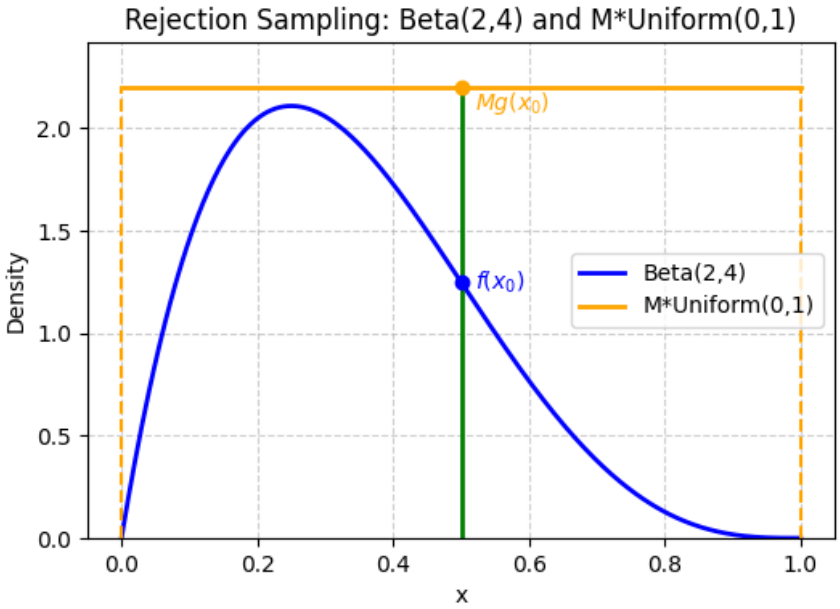


Figure 3.5: The sample x_0 and the evaluation line

The mathematical criteria to decide whether to accept or reject a sample involves randomness. Consider the green evaluation line that cross the points $f(x_0)$ and $Mg(x_0)$: if we are able to get a number on the evaluation line and it falls below $f(x_0)$ we will accept it, otherwise reject it. The idea is that the portions of the evaluation line below and above $f(x_0)$ are not equal and reflect the chances of occurrence of the sample in the distribution $f(x)$. To get this number we will draw from $Uniform(0, Mg(x))$. Thus, for a sample x_0, the mathematical criteria is to draw u from $Uniform(0, Mg(x_0))$ and accept x_0 if $u \leq f(x_0)$.

This is equivalent to:

1. sample x_0 from the distribution $g(x)$
2. draw u from the distribution $Uniform(0,1)$
3. accept x_0 if $u \leq \dfrac{f(x_0)}{Mg(x_0)}$

We could also see it as a simulation of a Bernoulli trial $\mathcal{B}(p)$ with parameter $p = \frac{f(x)}{Mg(x)}$: we accept the sample x_0 if $\mathcal{B}(p) = 1$. This is the reason why, in the signature algorithm of Figure 3.1, the output $(\mathbf{z}, \mathbf{c})$ is given with probability $min\left(\dfrac{D^m_\sigma(\mathbf{z})}{M \cdot D^m_{\mathbf{Sc},\sigma}(\mathbf{z})}, 1\right)$.

In our example, to visualize the output of the rejection sampling procedure we can do an experiment and sample 1000 times $x_1, ..., x_{1000}$ and see that the pairs (x_i, u_i) are

uniformly distributed over the area below the $Mg(x)$ and since we reject samples in the area above $f(x)$, our accepted samples are uniformly distributed over the area under $f(x)$. The results of this experiment are depicted in Figure 3.6.

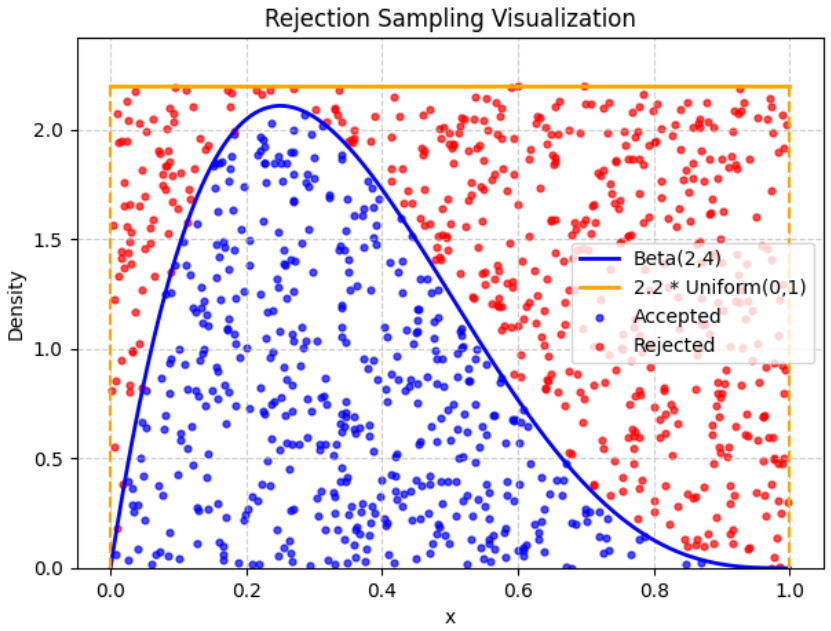


Figure 3.6: Rejection sampling visualization

We can see that many samples have been rejected, so it's natural to think how can we increase the efficiency of the rejection sampling algorithm. There are basically two ways to increase efficiency: we can choose a "better" proposal distribution $g(x)$ so that the rejection region is considerably smaller than the acceptance region or we can keep the same proposal distribution, but reduce the value of M.

Indeed, the quantity $\frac{1}{M}$ is the expected acceptance rate of the rejection sampling algorithm, meaning that the rejection sampling algorithm is expected to output an accepted sample within M iterations of the algorithm. In the example in Figure 3.6 the expected acceptance rate is $\frac{1}{M} = \frac{1}{2.2} = 0.455$, while the acceptance rate of the experiment is ≈ 0.473.

3.2.3 Why does rejection sampling work?

Following the standard proof in [10], we now verify that rejection sampling does return samples that follow the target distribution $f(x)$.

Let x, z be random variables with PDFs $f(x), g(z)$ and assume that we can sample from $g(z)$, and let b be a new binary random variable that represents the event that a sample is accepted (1) or rejected (0) such that $P(b = 1 \mid z) = \frac{f(z)}{Mg(z)}$. This is a Bernoulli trial with success parameter $\frac{f(z)}{Mg(z)}$. Our goal is to show that for any event $\mathcal{S}$, $P(z \in \mathcal{S} \mid b = 1) = P(x \in \mathcal{S})$, that is if we keep the accepted samples, their distribution matches $f(x)$

Using Bayes' theorem we can write:

$$P(z \in \mathcal{S} \mid b = 1) = \frac{P(b = 1 \mid z \in \mathcal{S}) \cdot P(z \in \mathcal{S})}{P(b = 1)}$$

Now,

- the numerator term $P(z \in \mathcal{S}) = \int_S g(z)dz$ because $z \sim g(z)$
- $P(b = 1 \mid z) = \frac{f(z)}{Mg(z)}$ because it's a Bernoulli
- the numerator term $P(b = 1 \mid z \in \mathcal{S})$ is not constant over $\mathcal{S}$, so we integrate $\int_S P(b = 1 \mid z)g(z)dz = \int_S \frac{f(z)}{Mg(z)}g(z)dz = \frac{1}{M}\int_S f(z)dz = \frac{1}{M}P(x \in \mathcal{S})$ because $\int_S f(z)dz$ is the probability under the target distribution
- the denominator $P(b = 1)$, marginalizing over z, equals $\int_Z P(b = 1 \mid z)g(z)dz = \int_Z \frac{f(z)}{Mg(z)}g(z)dz = \frac{1}{M}\int_Z f(z)dz = \frac{1}{M}$

Putting everything together we obtain

$$P(z \in \mathcal{S} \mid b = 1) = \frac{\frac{1}{M}P(x \in \mathcal{S})}{\frac{1}{M}} = P(x \in \mathcal{S})$$

This proves that the accepted samples have exactly the target distribution f(x).

3.2.4 Rejection sampling in lattice-based cryptography

As explained in "Lattice Signatures and Bimodal Gaussian" [11], rejection sampling was introduced into lattice-based constructions by Lyubashevsky in the context of identification schemes [12]. In an identification scheme the commitment message y in the first round of the protocol is used to hide the secret key s when computing the response z in the third round of the protocol. In classical cryptography identification schemes -like the Schnorr identification scheme- all operations take place in a finite ring, thus y being uniformly random hides s.

In a lattice-based scenario the secret key is a short vector s, then we need to choose y from a narrow distribution for the sake of correctness of the lattice-based primitives. However, since responses take the form of $z = y + cs$, their distribution $g(x)$ is correlated with the secret key s, which could potentially leak information about s.

Thanks to the rejection sampling algorithm, the distribution of the accepted responses would follow a fixed distribution $f(x)$ that doesn't depend on s. In this case we can say that the distribution of the accepted responses is statistically independent of the distribution of the secret key.

3.3 Rejection sampling as a matter of independence

Let $S \sim Unif\{-1,0,1\}$ and $Y \sim Unif\{-10,-9,...,9,10\}$ be discrete uniform random variables.

Consider the following simple protocol:

1. pick $s \leftarrow S$
2. pick $y \leftarrow Y$
3. compute $z = y + s$

We can make some observations:

- $P(z = 0 \mid s = -1) = \frac{1}{21}$
- $P(z = 0 \mid s = 0) = \frac{1}{21}$
- $P(z = 0 \mid s = 1) = \frac{1}{21}$
- $P(z = 9 \mid s = -1) = \frac{1}{21}$
- $P(z = 9 \mid s = 0) = \frac{1}{21}$
- $P(z = 9 \mid s = 1) = \frac{1}{21}$
- $P(z = 10 \mid s = -1) = 0$
- $P(z = 10 \mid s = 0) = \frac{1}{21}$
- $P(z = 10 \mid s = 1) = \frac{1}{21}$
- $P(z = 11 \mid s = -1) = 0$
- $P(z = 11 \mid s = 0) = 0$
- $P(z = 11 \mid s = 1) = \frac{1}{21}$

We could go over and over with all the possibilities, but the point is that we observe that if $z = 10$ we are sure that $s \neq -1$, and if we see $z = 11$ we are sure that $s = 1$. Thus, there are some responses that leak information about the value of the secret s.

The PMF of Z in figure 3.7 is the discrete trapezoid:

$$\mathbb{P}(Z = z) = \begin{cases} \frac{1}{63}, & z \in \{-11, 11\}, \\ \frac{2}{63}, & z \in \{-10, 10\}, \\ \frac{3}{63}, & z \in \{-9, -8, \ldots, 8, 9\}, \\ 0, & \text{otherwise.} \end{cases}$$

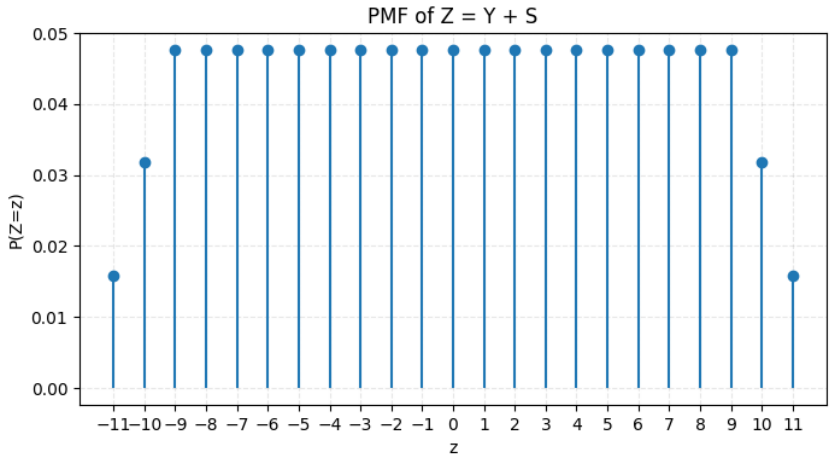


Figure 3.7: PMF of $Z = Y + S$

Now we do an experiment where we use rejection sampling to obtain responses z that don't leak anything about s. Our proposal distribution $g(x)$ is the discrete trapezoid, while as target distribution $f(x)$ we can choose the discrete uniform distribution $Uniform\{-9, -8, ...8, 9\}$ because we know that if $-9 \leq z \leq 9$ then no information about s is leaked and we know that it does not depend on the secret s.

The PMF of a discrete uniform is $f(x) = 1/n$, where n is the number of possible values in the distribution. The constraint on the value of M is $M \geq \frac{f(x)}{g(x)} = \frac{1/19}{1/21} = 1.105,\ \forall x \in \{-9, -8, ...,8,9\}$, so we choose $M = 1.11$.

The results of the experiment -using 2000 samples- are depicted in Figure 3.8. Please note that although the proposal and the target distributions are discrete by definition, we connected the probability mass points with lines for the sake of visualization.

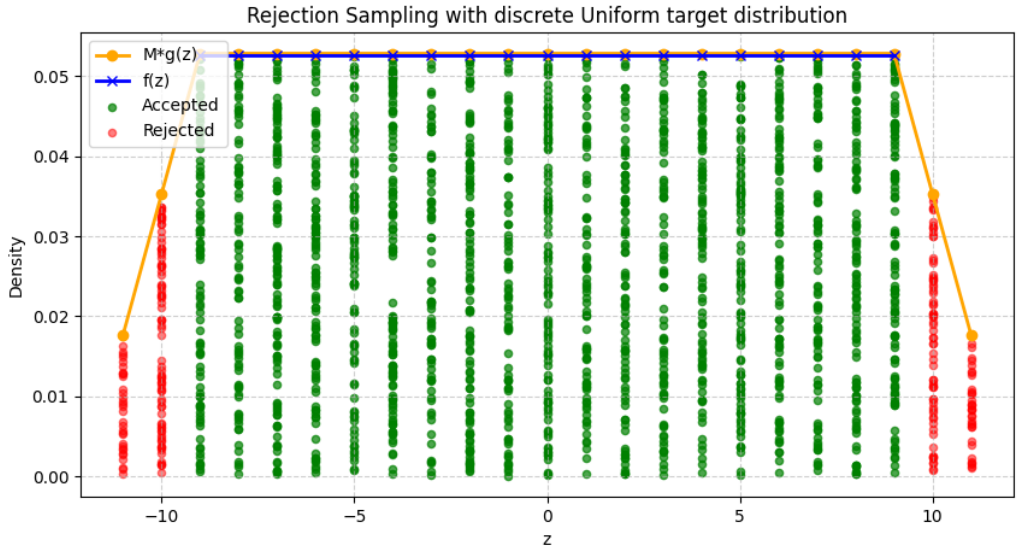


Figure 3.8: Rejection sampling with discrete Uniform as target distribution

The plot is not surprising: the accepted samples z fall in the acceptance region $[-9, ...,9]$, while the rejected samples fall outside of it, then the distribution of the output of our simple algorithm will follow a target distribution that doesn't depend on the secret s, achieving independence from the distribution of the secret s.

3.3.1 A simple indistinguishability experiment

The concept of independence can also be expressed in terms of indistinguishability.

Suppose that there are three different algorithms A_1, A_2, A_3 whose output may depend on some secret and:

- A_1 outputs are distributed according to a distribution g_1, that is the $Uniform(-8,8)$ distribution
- A_2 outputs are distributed according to a distribution g_2, that is the truncated Normal distribution $N(0, \sigma^2 = 4.5^2)$ over [-8,8]
- A_3 outputs are distributed according to a distribution g_3, that is the $Triangular(-8,8,0)$ distribution.

.

The normalized PDFs of g_1, g_2, g_3 over $[-8,8]$ are reported below.

$$g_1(x) = \begin{cases} \frac{1}{16}, & -8 \leq x \leq 8, \\ 0, & \text{otherwise.} \end{cases}$$

$$g_2(x) = \begin{cases} \frac{1}{0.924} \cdot \frac{1}{4.5 \cdot \sqrt{2\pi}} e^{-\frac{x^2}{2 \cdot 4.5^2}}, & -8 \leq x \leq 8, \\ 0, & \text{otherwise.} \end{cases}$$

$$g_3(x) = \begin{cases} \frac{x+8}{64}, & -8 \leq x \leq 0, \\ \frac{8-x}{64}, & 0 \leq x \leq 8, \\ 0, & \text{otherwise.} \end{cases}$$

In the first part of the experiment we suppose to collect 2000 output samples from each distribution g_i and plot their distributions as shown in Figure 3.9. As one would expect, the empirical distribution of the output samples drawn from g_i aligns closely to the theoretical PDF of g_i, possibly leaking information on the secret.

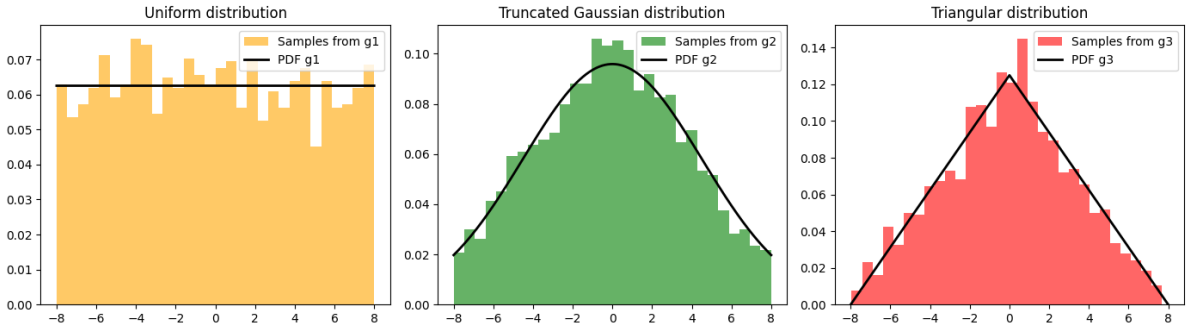


Figure 3.9: Distributions of the output samples for the proposal distributions.

In the second part of the experiment we suppose that the algorithms A_1, A_2, A_3 now implement the (classical) rejection sampling technique in the following way:

1. sample x from the distribution $g_i(x)$
2. sample u from the distribution $Uniform(0, Mg_i(x))$
3. accept x if $u \leq f(x)$

So the proposal distributions for each algorithm are respectively g_1, g_2, g_3, while the target distribution f is chosen to be the raised cosine distribution with parameters $\mu = 0, s = 8$, scaled by 5, that is $f(x) = 5 + 5 \cdot \cos\left(\frac{x\pi - 0}{8}\right)$ over [-8,8]. The normalized PDF of f is reported below.

$$f(x) = \begin{cases} \frac{1}{16} \cdot (1 + \cos\left(\frac{x\pi}{8}\right)), & -8 \leq x \leq 8, \\ 0, & \text{otherwise.} \end{cases}$$

We can see in Figure 3.10 the scaled three proposal distributions for $M_1 = 2.05, M_2 = 1.35, M_3 = 1.2$ respectively, while in Figure 3.11 are depicted the distributions of the output-samples of the algorithms when applying the rejection sampling technique.

By applying the rejection sampling technique, the distributions of the output samples follow the target distribution, thus deciding which algorithm has produced which distribution as well as trying to recover information about a potential secret becomes a very difficult task. That's why rejection sampling is used in cryptography to achieve independence between distributions.

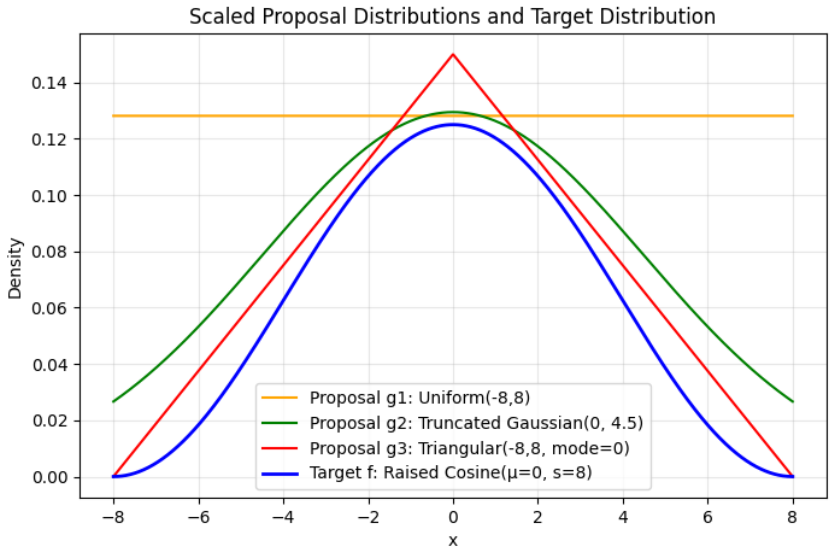


Figure 3.10: PDFs of the scaled proposal and target distributions of the experiment.

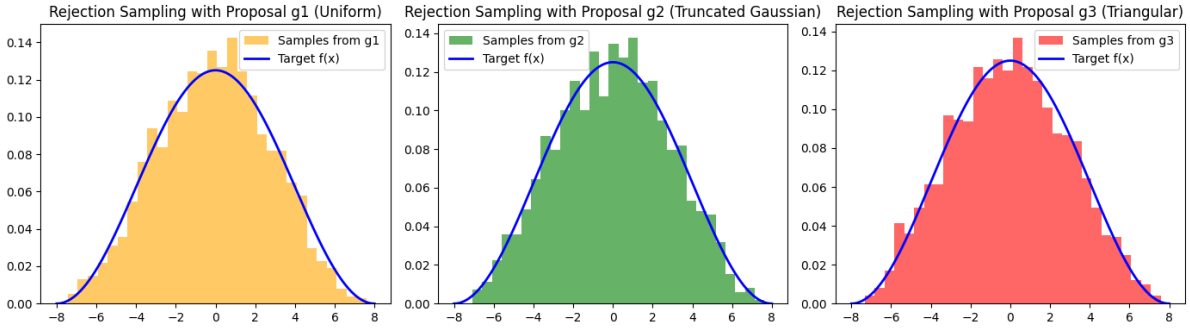


Figure 3.11: Distributions of the output samples for the proposal distributions when rejection sampling technique is adopted.

3.4 Rejection sampling with m-dimensional discrete Normal distribution

In this section we will see more in details the signature scheme of Figure 3.1 and the rejection sampling technique with discrete Normal distribution employed in such scheme.

The goal of the signature was to «come up with a distribution f and a distribution D such that for all $\mathbf{x}$ two properties are satisfied» [2, Section 1.2]:

- there is a **small** constant M such that $f(\mathbf{x}) \leq Mg(\mathbf{x})$, where g is the distribution of $\mathbf{z} = \mathbf{y} + \mathbf{Sc}$ for some random $\mathbf{c}$, where $\mathbf{y} \leftarrow D$
- the expected value of vectors distributed according to f is as small as possible, that is $\mathbb{E}(\mathbf{z}) \approx 0$

To satisfy those requirements Lyubashevsky chooses f, D to be the discrete m-dimensional Normal distribution D_σ^m with std $\sigma = \tilde{\Theta}(T) = \tilde{\Theta}(\sqrt{m})$ where $T = max||\mathbf{Sc}||$, already defined in Definition 3.2, and require that $f(\mathbf{x}) \leq Mg(\mathbf{x})$ for some small constant M, for the $\mathbf{x}$ that are not too big.

The main idea behind the structure of the signing algorithm is to make the distribution of $\mathbf{z}$ independent of the secret key $\mathbf{S}$. The target distribution is $f(\mathbf{x}) = D_\sigma^m$, while the proposal distribution g is the shifted discrete m-dimensional Normal distribution $g(\mathbf{x}) = D^m_{\mathbf{Sc},\sigma}$, because the $\mathbf{z}$ are obtained by shifting $\mathbf{y}$ by a small quantity $\mathbf{Sc}$.

In particular, there are many possible discrete m-dimensional Normal distributions $g(\mathbf{x}) = D^m_{\mathbf{Sc},\sigma}$ depending on the value of $\mathbf{Sc}$. A priori we know that the largest shift is given when $||\mathbf{Sc}|| = T = max(||\mathbf{Sc}||)$, and for this reason Lyubashevsky consider this "worst-case" scenario when stating the main technical rejection sampling Theorem 3.10. We can easily visualize this situation in Figure 3.12 with the 1-dimensional Normal distribution considering $|v| = max|Sc|$.

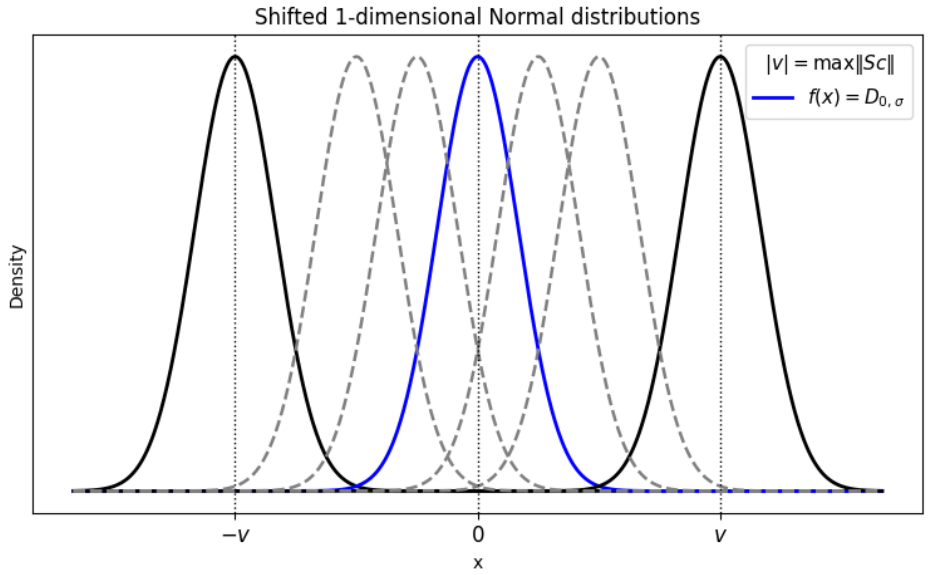


Figure 3.12: Shifted 1-dimensional Normal distributions

Intuitively, the greater the shift and the greater the value of M we will need to make sure that $Mg(x)$ envelops the target distribution $f(x)$. The advantage of working with Normal distributions is that if $\sigma \gg T$ then $f(x)$ and $g(x)$ are close distributions and it should not be too hard to find a small value M. We will see that the rejection sampling Theorem 3.10 shows that for appropriate values of M and σ, the signature algorithm will output a signature with probability approximately $\frac{1}{M}$ and the output's distribution is statistically close to the target distribution f.

This means that the expected length of the signature $\mathbb{E}(\mathbf{z}) \approx \sigma\sqrt{m} = \tilde{O}(m) = \tilde{O}(n)$. Thus, the length of the signatures is mainly affected by m, and lowering m will results in shorter signatures. Beware that we can't afford to just lower m while leaving everything else the same, because the $SIS_{q,n,m,d}$ search problem becomes easier (recovering $\mathbf{S}$ given $(\mathbf{A}, \mathbf{AS} \pmod q)$ becomes easier). We will see in section 3.6 how we can lower the value of m basing the security of our scheme on the other security assumptions defined in 3.1.1.

3.4.1 Facts about the discrete Normal distribution

Now we recall some facts from [2, Section 4] about the discrete m-dimensional Normal distribution and the main rejection sampling theorem.

Lemma 3.7 tells us that the inner product between an element $\mathbf{z} \sim D_\sigma^m$ and an element $\mathbf{v} \in \mathbb{R}^m$ is bounded.

Lemma 3.7. *For any vector* $\mathbf{v} \in \mathbb{R}^m$ *and any* $\sigma, r > 0$,

$$Pr[\ |\langle \mathbf{z}, \mathbf{v}\rangle| > r; \mathbf{z} \xleftarrow{\$} D_\sigma^m] \leq 2e^{-\frac{r^2}{2||\mathbf{v}||^2\sigma^2}}$$

Lemma 3.8.

1. *For any* $k > 0, Pr[\ |z| > k\sigma; z \xleftarrow{\$} D_\sigma^1] \leq 2e^{-\frac{k^2}{2}}$
2. *For any* $\mathbf{z} \in Z^m$ *and* $\sigma \geq 3/\sqrt{2\pi}$ *,* $D_\sigma^m(\mathbf{z}) \leq 2^{-m}$
3. *For any* $k > 1$, $Pr[\ ||\mathbf{z}|| > k\sigma\sqrt{m}\ ; \mathbf{z} \xleftarrow{\$} D_\sigma^m] < k^m e^{\frac{m}{2}(1-k^2)}$

Lemma 3.9 will be used to bound the success probability of the rejection sampling algorithm.

Lemma 3.9. *For any* $\mathbf{v} \in \mathbb{Z}^m$*, if* $\sigma = \omega(||\mathbf{v}||\sqrt{\log m})$*, then*

$$Pr[D_\sigma^m(\mathbf{z})/D_{\mathbf{v},\sigma}^m(\mathbf{z}) = O(1); \mathbf{z} \xleftarrow{\$} D_\sigma^m] = 1 - 2^{-\omega(\log n)},$$

and more specifically, for any $\mathbf{v} \in \mathbb{Z}^m$*, if* $\sigma = \alpha||\mathbf{v}||$ *for any positive* α*, then*

$$Pr[D_\sigma^m(\mathbf{z})/D_{\mathbf{v},\sigma}^m(\mathbf{z}) < e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}}; \mathbf{z} \xleftarrow{\$} D_\sigma^m] > 1 - 2^{-100}.$$

More specifically, from Lemma 3.7 we know that for $r = 12||\mathbf{v}||\sigma$, with probability at least $1 - 2^{-100}$, $|\langle \mathbf{z}, \mathbf{v}\rangle| \leq 12||\mathbf{v}||\sigma$, and therefore, with probability at least $1 - 2^{-100}$,

$$exp\left(\frac{-2\langle \mathbf{z}, \mathbf{v}\rangle + ||\mathbf{v}||^2}{2\sigma^2}\right) < exp\left(\frac{24||\mathbf{v}||\sigma + ||\mathbf{v}||^2}{2\sigma^2}\right) = e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}},$$

where the last equality uses $\sigma = \alpha||\mathbf{v}||$.

3.4.2 Rejection sampling theorem

The following theorem taken from [2, Theorem 4.6] is the formal guarantee that the rejection sampling trick works.

Theorem 3.10. *Let* V *be a subset of* $\mathbb{Z}^m$ *in which all elements have norm less than* T*,* σ *be some element in* $\mathbb{R}$ *such that* $\sigma = \omega(T\sqrt{\log m})$*, and* $h : V \to \mathbb{R}$ *be a probability distribution. Then there exists a constant* $M = O(1)$ *such that the distribution of the following algorithm* $\mathcal{A}$*:*

1. $\mathbf{v} \xleftarrow{\$} h$
2. $\mathbf{z} \xleftarrow{\$} D_{\mathbf{v},\sigma}^m$
3. *output* $(\mathbf{z}, \mathbf{v})$ *with probability* $min\left(\frac{D_\sigma^m(\mathbf{z})}{MD_{\mathbf{v},\sigma}^m(\mathbf{z})}, 1\right)$

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

1. $\mathbf{v} \xleftarrow{\$} h$

2. $\mathbf{z} \xleftarrow{\$} D_\sigma^m$

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

Moreover, the probability that $\mathcal{A}$ *outputs something is at least* $\dfrac{1 - 2^{-\omega(\log m)}}{M}$.

More concretely, if $\sigma = \alpha T$ *for any positive* α*, then* $M = e^{\frac{12}{\alpha} + \frac{1}{2\alpha^2}}$*, the output of algorithm* $\mathcal{A}$ *is withing statistical distance* $\frac{2^{-100}}{M}$ *of the output of* $\mathcal{F}$*,and the probability that* $\mathcal{A}$ *outputs something is at least* $\frac{1-2^{-100}}{M}$.

Let's try to understand what this theorem is telling us. First of all the vector $\mathbf{v} \in V \subset \mathbb{Z}^m$ represents the product $\mathbf{Sc}$, because V is the subset of all possible $\mathbf{Sc} \in Z^m$ where $T = max||\mathbf{Sc}||$.

Algorithm $\mathcal{A}$ is our algorithm that uses rejection sampling to make a sample $\mathbf{z}$ from a shifted discrete m-dimensional Normal distribution (centered at $\mathbf{v}$) look like it came from the centered discrete m-dimensional Normal distribution (centered at 0).

Algorithm $\mathcal{F}$ is the ideal behavior of our algorithm, and it outputs pairs $(\mathbf{z}, \mathbf{v})$ where $\mathbf{z} \sim D_\sigma^m$ regardless of $\mathbf{v}$. This is only theoretical as we can't sample $\mathbf{z}$ independently of $\mathbf{v}$ in our protocol.

So the theorem is telling us that if we use Algorithm $\mathcal{A}$, then the distribution of its output $(\mathbf{z}, \mathbf{v})$ is statistically close (with negligible distance) to the output of Algorithm $\mathcal{F}$. That is, rejection sampling "sanitize" the dependency of $\mathbf{z}$ from $\mathbf{v}$ and $\mathbf{z} \sim D_\sigma^m$.

3.4.3 Proof of security of the signature scheme

Before diving into the proof of security [2, Section 5], we underline the fact that we (the signer) are the one that first generate (randomly) both the secret key $\mathbf{S}$ and the matrix $\mathbf{A}$, and then compute $\mathbf{T} = \mathbf{AS} \pmod q$. Thus, an Adversary that tries to recover the secret key $\mathbf{S}$ given the public key $(\mathbf{A}, \mathbf{AS})$ must solve the $\text{SIS}_{q,n,m,d}$ search problem. We thought it was important to underline that because the security of the signature scheme is not about the hardness of recovering the secret key, but the hardness of forging valid signatures.

Let's provide the security reduction (see Figure 3.13) in case we base the security of the signature scheme on the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem. We show that an Adversary that is able to forge valid signatures can retrieve a non-trivial solution for the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem for $\beta \approx \widetilde{O}(||\mathbf{z}||)$.

The entities in the security reduction are:

- The Adversary: the one who is able to break the signature scheme and can query the signing oracle and the random oracle H.
- Signing oracle: the oracle that answers to the adversary's signing requests. Its behavior his depicted in Figure 3.14.
- Challenger: the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ instance that provides the matrix $\mathbf{A}$.

We won't go into the details of the theorems and lemmas that Lyubashevsky describes to formally prove the security of the signature scheme, but we will just give the high-level overview of the security reduction.

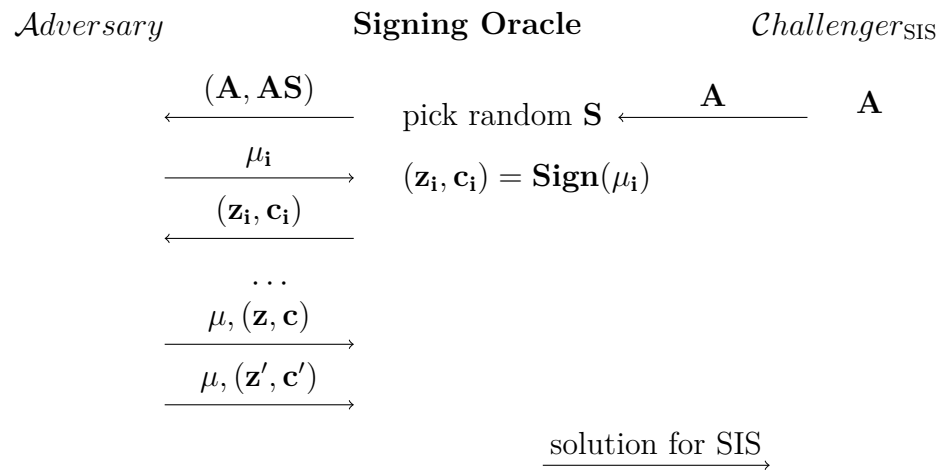


Figure 3.13: Security reduction for the signature scheme.

The idea is that given a matrix $\mathbf{A}$, we can create a secret key $\mathbf{S}$ and publish the public key $(\mathbf{A}, \mathbf{AS})$. Then, the Adversary picks messages μ_i and interacts with the Signing oracle to obtain the corresponding valid signatures pairs $(\mathbf{z_i}, \mathbf{c_i})$. How can the Signing oracle produce such signature pairs? Since we use the rejection sampling technique in the signing algorithm, we have that the signature pair $(\mathbf{z}, \mathbf{c})$ is independent of the secret key $\mathbf{S}$. Then, the Signing oracle can produce such signatures by generating randomly $\mathbf{c}, \mathbf{z}$ as in Figure 3.14 and programming the random oracle H accordingly. This wouldn't be possible if we had not obtained the independence of $\mathbf{z}$ from $\mathbf{S}$ thanks to rejection sampling.

$\mathbf{Sign}(\mu, \mathbf{A}, \mathbf{S})$

1: $\mathbf{c} \xleftarrow{\$} \{\mathbf{v} : \mathbf{v} \in \{-1,0,1\}^k, ||\mathbf{v}||_1 \leq \kappa\}$
2: $\mathbf{z} \xleftarrow{\$} D_\sigma^m$
3: with probability 1/M,
4: output $(\mathbf{z}, \mathbf{c})$
5: Program $H(\mathbf{Az} - \mathbf{Tc}, \mu) = \mathbf{c}$

Figure 3.14: Signing oracle

So at some point the Adversary will be ready to break the scheme and will output a valid message-signature pair for a message μ, and for the forking lemma he can output a second valid message-signature pair for the same message μ. Now we are able to extract a solution $\mathbf{v}$ for the SIS problem: the two signatures forged by the Adversary satisfy the verification equation and $\mathbf{z}, \mathbf{z}'$ are small, thus we can write $\mathbf{Az} = \mathbf{Tc}$ and $\mathbf{Az}' = \mathbf{Tc}'$. It follows that $\mathbf{A}(\mathbf{z} - \mathbf{z}' + \mathbf{Sc} - \mathbf{Sc}') = 0$, but we need that $\mathbf{v} = (\mathbf{z} - \mathbf{z}' + \mathbf{Sc} - \mathbf{Sc}')$ is $\neq 0$.

One important constraint to prove that $\mathbf{v} \neq \mathbf{0}$ is that there must be a second (unknown to us) valid secret key $\mathbf{S}' \neq \mathbf{S}$ such that $\mathbf{AS} = \mathbf{AS}'$ and the Adversary can not know which secret key we know. To satisfy this constraint, we require a particular relationship between the parameters *q, n and m* as in Lemma 3.11, while the indistinguishability of $\mathbf{S}$ and $\mathbf{S}'$ is satisfied because the distribution of the signature is independent of the secret key.

Lemma 3.11. *For any* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *where* $m \geq 64 + n \cdot (\log q / \log(2d+1))$*, for randomly chosen* $\mathbf{s} \xleftarrow{\$} \{-d, ..., 0, ..., d\}^m$*, with probability* $1 - 2^{-100}$*, there exists another* $\mathbf{s}' \xleftarrow{\$} \{-d, ..., 0, ..., d\}^m$ *such that* $\mathbf{As} = \mathbf{As}'$*.*

3.5 Analysis of the parameters

Consider the digital signature protocol in Figure 3.1. For the security parameter $\lambda = 100$, we consider distributions to be statistically close if they are $\approx 2^{-100}$ apart and we also require ≈ 100 bits of security from the cryptographic hash function H used in the signature scheme, because in general when we use Fiat-Shamir to obtain a signature scheme we only require the random oracle to output λ bits [2, Section 5] .

For the verification algorithm, since $\mathbf{z}$ will be distributed according to D_σ^m, for Lemma 3.8 with $k = \eta$ we know that with probability $1 - 2^{-100}$ we have $||\mathbf{z}|| < \eta\sigma\sqrt{m}$, and since $\mathbf{Az} = \mathbf{Az} - \mathbf{Tc}$, the signature will be accepted.

Now we present in Table 3.1 the parameters selected by Lyubashevsky in [2].

Parameter	I	II	III	IV	V
n	512	512	512	512	512
q	2^{27}	2^{25}	2^{33}	2^{18}	2^{26}
d	1	1	31	1	31
k	80	512	512	512	512
η	1.1	1.1	1.2	1.3	1.3
$m \approx 64 + \frac{n \log q}{\log(2d+1)}$	8786	8139	3253	–	–
			2891	–	–
$m = 2n$ (used in Sec. 3.6)	–	–	–	1024	1024
κ s.t. $2^\kappa \cdot \binom{k}{\kappa} \geq 2^{100}$	28	14	14	14	14
$\sigma \approx 12 \cdot d \cdot \kappa \cdot \sqrt{m}$	31495	15157	300926	–	–
			280024	–	–
$\sigma \approx 6 \cdot d \cdot \kappa \cdot \sqrt{m}$ (used in Sec. 3.6)	–	–	–	2688	83328
$M \approx \exp\left(\frac{12d\kappa\sqrt{m}}{\sigma} + \left(\frac{d\kappa\sqrt{m}}{2\sigma}\right)^2\right)$	2.72	2.72	2.72	7.4	7.4
approx. signature size (bits) $\approx m \log(12\sigma)$	163000	142300	73000	14500	19500
			62677		
approx. secret key size (bits) $\approx m \cdot k \cdot \log(2d+1)$	2^{20}	$2^{22.5}$	2^{23}	$2^{19.5}$	$2^{21.5}$
approx. public key size (bits) $\approx n \cdot k \cdot \log q$	2^{20}	$2^{22.5}$	2^{23}	$2^{22.1}$	$2^{22.7}$

Table 3.1: Parameters of the signature scheme

The parameters in columns I, II, III are based on the hardness of the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem where $\beta = (2\eta\sigma + 2d\kappa)\sqrt{m}$, while columns IV, V are based on the hardness of the $\text{SIS}_{q,n,m,d}$ problem. Moreover, the parameters in column V are also compatible with the LWE assumption that we will see in section 3.6.2. Please note that we checked the correctness of all the proposed parameters and we found that the values of m, σ and the signature size of the signature in column III are not correct. For this reason we computed

and highlighted in bold the correct values of those parameters.

3.5.1 Parameters that influence the SIS problems

The parameters n and q. These parameters are determined by the underlying SIS problem instance that we want to work with. The determinant of the full rank lattice is $\approx q^n$ and the larger it is, the harder is finding short vectors in the lattice.

The parameter β. If the underlying security assumption is the hardness of the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem, then we know from section 3.1.1 that we require $\beta \geq \sqrt{m}q^{n/m}$. For the columns I, II, II we assume $\beta = (2\eta\sigma + 2d\kappa)\sqrt{m}$ [3]. The values of β and the respective lower bound for each column are represented in Table 3.2.

	I	II	III
lower bound $= \sqrt{m}\, q^{n/m}$	278.96	268.35	2087.78
$\beta = (2\eta\sigma + 2d\kappa)\sqrt{m}$	6.50×10^6	3.01×10^6	4.12×10^7

Table 3.2: Lower bounds and values of β in I, II, III

The parameter d. The parameter d bounds the absolute value of the coefficients of the secret key $\mathbf{S}$. For the columns I,II,III we don't have any constraint on d, but for the columns IV, V, where the security assumption is the hardness of the $\text{SIS}_{q,n,m,d}$ problem we required from section 3.1.1 that $d \ll q^{n/m}$. The values of d and the respective upper bound for column IV and V are represented in Table 3.3.

	IV	V
upper bound $= q^{n/m}$	512	8192
d	1	31

Table 3.3: Upper bounds and values of d in IV, V

The parameter m. The parameter m is determined by the underlying SIS problem instance that we want to work with. For columns I, II, III the Lemma 3.11 bounds $m \geq 64 + \frac{n \log q}{\log(2d+1)}$, while we will set $m = 2n$ when dealing with columns IV, V. Remind that m is the parameter that mostly impact the signature size as $\mathbb{E}(\mathbf{z}) \approx \sigma\sqrt{m}$, thus the lower it is the shorter the signatures will be.

3.5.2 Parameters that influence the signature protocol

The parameter η. This parameter is used to bound the norm of $\mathbf{z}$. As $\mathbf{z} \sim D_\sigma^m$ and we know that the expected value of D_σ^m is about $\sigma\sqrt{m}$, Lemma 3.8 with $k = \eta$ tells us that $||\mathbf{z}|| < \eta\sigma\sqrt{m}$ with probability $1 - 2^{-100}$.

[3] This formula is set by Lyubashevsky in one of the theorem used for the security proof of the signature scheme in [2].

The parameters k and κ. The parameter k determines the length (in bits) of the challenge vector returned by the cryptographic hash function H, while the parameter κ determines the maximum number of coefficients of the challenge vector that are non-zero. Now, the number of possible challenges determines the soundness error $\epsilon = \frac{1}{\#challenges}$ and in order to achieve a security level of λ in 1 round of the protocol we need that $\log(\frac{1}{\epsilon}) \geq \lambda$, that is $\#challenges \geq 2^\lambda$.

The challenge space is $\mathcal{C}_{k,\kappa} = \{\mathbf{c} \in -1{,}0{,}1^k : ||\mathbf{c}||_\mathbf{1} \leq \kappa\}$, and the number of vectors in such space is given by

$$|\mathcal{C}_{k,\kappa}| = \sum_{i=0}^{\kappa} \binom{k}{i} \cdot 2^i.$$

To lower bound this value, Lyubashevsky says

$$|\mathcal{C}_{k,\kappa}| \geq 2^\kappa \cdot \binom{k}{\kappa}.$$

Thus, for the security level $\lambda = 100$, we obtain the constraint on the choice of κ (also shown in Table 3.1):

$$\kappa : \ 2^\kappa \cdot \binom{k}{\kappa} \geq 2^{100}$$

The parameters M and σ. The parameter M is the expected number of iterations of the rejection sampling algorithm, while σ is the std of the discrete m-dimensional Normal distribution and it impacts on the signature's length as well as on the parameter M.

We know that Lemma 3.9 gives an upper bound for

$$M \geq \frac{f(x)}{g(x)} = \frac{D_\sigma^m(\mathbf{z})}{D_{\mathbf{v},\sigma}^m(\mathbf{z})}$$

In particular, considering $\mathbf{v} = \mathbf{Sc} \in \mathbb{Z}^m$ and $\sigma = \alpha||\mathbf{Sc}||$, we choose $M = e^{\frac{12}{\alpha} + \frac{1}{2\alpha^2}}$.

For columns I, II, III we set $\alpha = 12$ to obtain $M = e \approx 3$ and $\sigma = 12||\mathbf{Sc}||$. We now see how σ can be approximated to $\sigma \approx 12 \cdot \kappa \cdot d \cdot \sqrt{m}$.

We recall that $\mathbf{S} \in \{-d, ..., 0, ...d\}^{m \times k}$ and $\mathbf{c} \in \{-1{,}0{,}1\}^k$ with $||\mathbf{c}||_1 \leq \kappa$.

The vector $\mathbf{Sc}$ is just a linear combination of some columns of $\mathbf{S}$

$$\mathbf{Sc} = \sum_{i:c_i \neq 0} c_i \mathbf{S_i},$$

where each $\mathbf{S_i}$ is a column of $\mathbf{S}$. Since $\mathbf{c}$ has at most κ non-zero entries, the sum involves at most κ columns. Each column $\mathbf{S_i}$ is an m-dimensional vector with entries in $[-d, d]$, then its squared Euclidean norm is bounded by

$$||\mathbf{S_i}||^2 = \sum_{j=1}^{m} \mathbf{S}_{j,i} \leq m \cdot d^2.$$

Thus, $||\mathbf{S_i}|| \leq d\sqrt{m}$.

Using the triangle inequality:

$$||\mathbf{Sc}|| = || \sum_{i,c_i \neq 0} c_i \mathbf{S_i} \ || \leq \sum_{i:c_i \neq 0} ||\mathbf{S}_i|| \leq \kappa \cdot max||\mathbf{S_i}|| \leq \kappa \cdot d \cdot \sqrt{m}$$

Then, $T = max||\mathbf{Sc}|| = \kappa \cdot d \cdot \sqrt{m}$, and $\sigma \approx 12 \cdot T = 12 \cdot \kappa \cdot d \cdot \sqrt{m}$ from Theorem 3.10.

For columns IV,V $\sigma \approx 6 \cdot \kappa \cdot d \cdot \sqrt{m}$ and M ≈ 7.4. It's easy to check that in this case $\alpha = 6$.

3.5.3 Signature size and key size

- The secret key $\mathbf{S}$ is a $m \times k$ matrix with coefficients of maximum absolute value d. Then, it can be represented by $mk \cdot \log(2d+1)$ bits.
- The public key $(\mathbf{A}, \mathbf{T})$ can be split in two parts: the matrix $\mathbf{A}$ can be shared among all users, while the matrix $\mathbf{T} \in \mathbb{Z}_q^{n \times k}$ is individual and requires $nk \cdot \log q$ bits.
- The signature pair $(\mathbf{z}, \mathbf{c})$ is dominated by $\mathbf{z}$, and for Lemma 3.8 (1) the length of each coefficient of $\mathbf{z}$ is at most 12σ with probability at least $1 - 2^{-100}$. Thus, $\mathbf{z}$ can be represented with approximately $m \cdot \log(12\sigma)$.

3.5.4 Lowering the parameter the signature's size

There are some "base" parameters that we must decide at first, and some "dependent" parameters that depend on the others. The "base" parameters in this sense are n, q, d, k, α, while the "dependent" parameters are m, κ, σ, M,.

We can see in Table 3.4 how the "dependent" parameters are influenced by the others. Then we will see that there some trade-offs when setting the parameters, in particular between the signatures size and the key sizes.

Parameter	**Growth ↑ behavior**
m	linear in n; $\log q$; $1/\log(2d)$
κ	combinatorial in k
σ	linear in d, κ, α; $\sqrt{m}$
M	inverse in α
signature $\mathbf{z}$ size	linear in m; $\log(12\sigma)$
secret key $\mathbf{S}$ size	linear in m, k; $\log(2d)$
public key $\mathbf{T}$ size	linear in n, k; $\log q$

Table 3.4: Dependencies among the parameters

Let's see in which ways we can decrease the signature size.

Increasing k. Consider the values of the parameters from column I to column II of Table 3.1.

- The value of k has been increased from $k_I = 80$ to $k_{II} = 512$, that is 6.4 times the initial value.
- This allow us to lower the value of κ from $\kappa_I = 28$ to $\kappa_{II} = 0.5\kappa_I = 14$.
- Thus, the value of σ is halved and the we expect a reduction of the signature size.

In particular, $\sigma_{II} = 0.5\sigma_I$ and $\log(12\sigma_{II}) = \log(6\sigma_I)$, then $\dfrac{\log(6\sigma_I)}{\log(12\sigma_I)} = 0.946$, which means that the size of the signature in column II will be $\approx 95\%$ of the size of the signature in column I.

Actually, if we look carefully at column II, we will notice that the values of q and m have been slightly decreased... then the signature size in column II is $\approx 90\%$ of the

signature size in column I. The trade-off is that the key sizes in column II would be 6.4 times larger that the key sizes in column I.[4]

Increase d. A different approach could be to increase the value of d to decrease the value of m. Of course σ would also increase because it grows linearly with d and only proportionally to the square root of m, but the signature size grows linearly with m and only logarithmically with σ, so it should not be a problem.

Consider the columns II and III of Table 3.1.

- The value of d has been increased from $d_{II} = 1$ to $d_{III} = 31$, while q has been increased from $q_{II} = 2^{25}$ to $q_{III} = 2^{33}$
- This results in a reduction of the value of m from $m_{II} = 8139$ to $m_{III} = 2891$, that is a reduction of $\approx 64{,}5\%$,
- but σ increases from $\sigma_{II} = 15157$ up to $\sigma_{III} = 280024$, that is an increase of $\approx 1847\%$

However, the signature size is reduced from 142300 bits to 62677 bits, that is a reduction of $\approx 66\%$. On the other hand, the keys size has slightly increased from $\approx 2^{22.5}$ bits to $\approx 2^{23}$ bits.

3.6 Basing the scheme on low-density SIS and LWE

Consider the parameters in column III of Table 3.1 and suppose to lower d from 31 down to 1, without changing m accordingly, that is $m \ll 64 + n \cdot \log q/(\log 2d + 1)$. This modification decreases σ by a factor of d, which causes the signatures vector $\mathbf{z}$ to be smaller. Thus, the problem of forging signature vectors $\mathbf{z}$ becomes harder because now the Adversary must find smaller signatures vector $\mathbf{z}$. Then, we would be able to lower other parameters such as q and m, further reducing the length of the signature.

This looks great, but there is an issue: since $m \ll 64 + n \cdot \log q/(\log 2d + 1)$ in the first place, the security proof that we used is not valid anymore, because with very high probability $\forall \mathbf{T}\ \exists!\ \mathbf{S} : \mathbf{AS} = \mathbf{T}$. Indeed, we can easily check that if we plug into $d = 1 \ll q^{n/m}$ the values of q, n, m taken from column III, we obtain $1 \ll 55.91$, that is we are in low-density $SIS_{q,n,m,d}$ region.

Even though the security proof loses this important constraint, Lyubashevsky comes up with a clever observation to still be able to exploit the security reduction: the Signing oracle in Figure 3.14 does not use the secret key $\mathbf{S}$ and its output is indistinguishable from the real signature algorithm. Then the idea is, in the security proof, to pick a secret key $\mathbf{S}'$ with large coefficients -instead of our real secret key $\mathbf{S}$- so that there will exist another secret key $\mathbf{S}''$ such that $\mathbf{AS}' = \mathbf{AS}''$.

In the security proof in section 3.4.3 the Adversary is not able to distinguish between the signing algorithm that uses $\mathbf{S}$ and the Signing oracle. Now the issue could be that the Adversary is able to distinguish them because the signing algorithm is using a different key $\mathbf{S}'$. However, if distribution of $(\mathbf{A}, \mathbf{AS})$ is indistinguishable from the distribution of $(\mathbf{A}, \mathbf{AS}')$ (and it is, based on the hardness of the low-density $SIS_{q,n,m,d}$ problem from

[4]Actually $\approx 2^{22.5} = 5.6$ times larger because of the slightly reduced values of q and m that affect, respectively, the public key size and the secret key size.

definition 3.6), then the adversary $\mathcal{A}$ would not notice that he is given an invalid key-pair. Moreover, as the Signing oracle doesn't use any secret key when answering to the Adversary, the Adversary would still behave in the same way and find a solution for the ℓ_2-SIS$_{q,n,m,\beta}$ problem.

Thus, our signature scheme is based on the hardness of two problems: the low density $SIS_{q,n,m,d}$ problem (for the hardness finding the secret key) and the ℓ_2-SIS$_{q,n,m,\beta}$ problem with $\beta = (2n\sigma+2d'\kappa)\sqrt{m}$ (for the hardness of forging signatures), where d' is the maximum absolute value for the coefficients of the secret key $\mathbf{S}'$ used in the proof.

In his work [2, Section 6] Lyubashevsky also gives some bounds to make sure that those problems are hard. In particular:

- for the $SIS_{q,n,m,\beta}$ problem we require $\beta < min(q, 2^{2\cdot\sqrt{n\cdot\log q\cdot\log\delta}})$, where $\delta = 1.007$ [5]
- for the $SIS_{q,n,m,d}$ decision problem we require that $\frac{\beta\psi}{q} \geq 2$ where $\psi = \sqrt{d(d+1)m/3}$

3.6.1 The LWE problem

Definition 3.12 (LWE$_{q,n}$ **distribution**). *Choose randomly a vector* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ *and a small error vector* $e \xleftarrow{\$} \mathbb{Z}_q$. *The LWE-distribution is defined as* $A_{\mathbf{s}} = \{(\mathbf{a}, b)\} \subseteq \mathbb{Z}_q^n \times \mathbb{Z}_q$ *such that:*

- $\mathbf{a} \in \mathbb{Z}_q^n$ *is randomly sampled by a uniform distribution*
- $b = \langle \mathbf{s}, \mathbf{a} \rangle + e \pmod q$

Definition 3.13 (LWE$_{q,n,m}$ **search problem**). *Find* $\mathbf{s} \in \mathbb{Z}_q^n$ *given* m *pairs* $(\mathbf{a}, b) \in A_{\mathbf{s}}$.

Definition 3.14 (LWE$_{q,n,m}$ **decision problem**). *Given* m *pairs* $(\mathbf{a}, b)$ *tell if they belong to a LWE distribution (with the same fixed* $\mathbf{s} \in \mathbb{Z}_q^n$ *or if they are randomly sampled from a uniform distribution in* $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The LWE search problem and the LWE decision problem are computationally equivalent. It's possible to define an equivalent version of the LWE problem where the secret key is not selected from the uniform distribution $\mathbb{Z}_q^n$, but is selected from the Normal distribution D_ψ^n.

3.6.2 Low density SIS is equivalent to the LWE problem

At this point, Lyubashevsky makes the following observation [2, Section 6.1]: if we consider a matrix $\mathbf{A} = [\bar{\mathbf{A}}||\mathbf{I}] \in \mathbb{Z}_q^{n\times 2n}$, where $\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times n}$, then the problem of distinguishing pairs $(\mathbf{A}, \mathbf{As})$, where $\mathbf{s} \xleftarrow{\$} D_\psi^{2n}$, from uniformly distributed pairs in $\mathbb{Z}_q^{n\times 2n} \times \mathbb{Z}_q^{2n}$ is as hard as LWE.

Now, considering a $2n \times k$ matrix $\mathbf{S}$ where each of the k columns is distributed according to D_ψ^{2n} we could say that distinguishing pairs $(\mathbf{A}, \mathbf{AS})$ from uniformly distributed pairs in in $\mathbb{Z}_q^{n\times 2n} \times \mathbb{Z}_q^{2n\times k}$ is as hard as LWE.

[5] It's easy to check that the parameters in columns I, II, III respect this bound

Then, the LWE problem is exactly the low-density $SIS_{1,n,2n,d}$ problem, except for the distribution of the secret key $\mathbf{S}$, and so we can base the security of the signature scheme on the hardness of LWE problem rather than the hardness of low density SIS problem.

The thing we need to care about is the norm of each of the columns of $\mathbf{S}$. If the norm of $\mathbf{s} \xleftarrow{\$} D_\psi^m$ is approximately the norm of $\mathbf{s}' \xleftarrow{\$} \{-d, ...0, ..., d\}^m$, then we don't really need to change or modify the security and correctness of the signature scheme that we have seen so far. In particular, if $\psi \approx \sqrt{\frac{d(d+1)}{3}}$, then $||\mathbf{s}|| \approx ||\mathbf{s}'||$ because $||\mathbf{s}||$ concentrated around $\psi\sqrt{m}$ while $||\mathbf{s}'||$ is concentrated around $\sqrt{d(d+1)m/3}$.

Thus, the signature size and key sizes in column V of Table 3.1 are compatible with the LWE assumption for $\psi \approx 18$.

3.6.3 The security advantage of basing the hardness of finding the secret on LWE or low-density SIS

The motivation behind the choice of basing the hardness of finding the secret key on LWE (or low-density SIS) rather than on SIS are discussed very well by Lyubashevsky in the talk at Microsoft in 2016 [6]. In the talk, Lyubashevsky says that lattice problems like LWE and SIS are just modern versions of the Knapsack problem over vectors, a popular problem in the '80s. Very briefly the setting of the Knapsack problem is the following:

- the secret key is a small norm vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^m$
- the public key is the pair $(\mathbf{A}, \mathbf{t})$ where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{t} = \mathbf{As} \pmod q$
- the problem: given $(\mathbf{A}, \mathbf{t})$ find $\mathbf{s}'$ such that $\mathbf{As}' = \mathbf{t} \pmod q$

The hardness of the Knapsack problem can be visualized in Figure 3.15. The problem is easy at the ends of the graph because:

- if $||\mathbf{s}||$ is very low, then there are only few possible vectors that solves the problem, in particular the ones with few non-zero components and so we just try them out until we find a solution
- if $||\mathbf{s}||$ is very high, then there are many possible vector that solves the problem, and it's not hard to find a solution via gaussian elimination

The problem is the hardest at the point P, when the entropy of $\mathbf{s}$ is approximately the same as the entropy of $\mathbf{t}$. In that point, we will have almost exactly one solution.

Thus, we want to build schemes that are based on problems that are hard around that point P. In particular, Lyubashevsky defines the two regions of SIS problem and LWE problem, that we could also think as high-density SIS[6] and low-density SIS.

We have already seen in section 3.6.2 how basing the scheme on LWE allow us to obtain a more efficient signature scheme in term of signature size, but there is one more security-related advantage that we are going to explain now.

Consider the Figure 3.16. When we base the security of the signature scheme on the ℓ_2-$\text{SIS}_{q,n,m,\beta}$ problem only, we put a condition on the secret key, that is that there must

[6] In this region there are many $\mathbf{s}$ such that $\mathbf{As} = \mathbf{t}$, then for classic ℓ_2-SIS problem we have $\mathbf{t} = \mathbf{0}$.

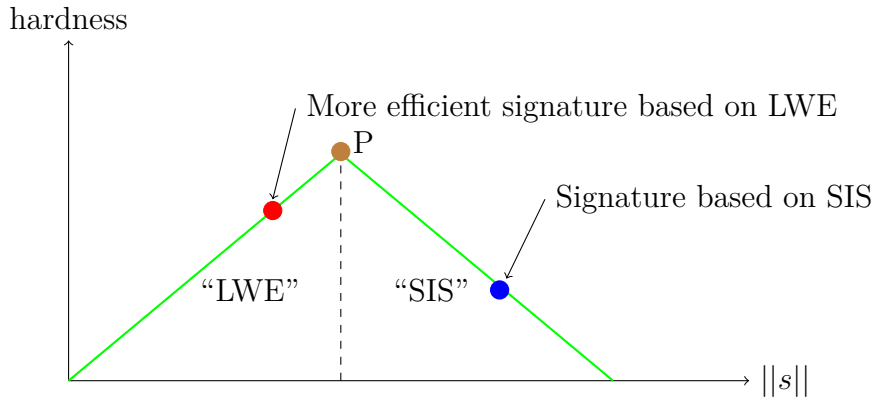


Figure 3.15: Hardness of Knapsack problem.

exists at least two different $\mathbf{S}, \mathbf{S}'$ such that $\mathbf{AS} = \mathbf{AS}'$. This situation is represented by the blue circle (1), located just a little to the right of the point P because we require (at least) one collision to exist.

On the other hand, in the security proof of section 3.4.3, we have $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$ and the vector that solves SIS is $\mathbf{v} = (\mathbf{z} - \mathbf{z}' + \mathbf{Sc} - \mathbf{Sc}')$. Then, since $||\mathbf{z}|| > ||\mathbf{S}||$, the vector $\mathbf{v}$ is dependent of $\mathbf{z}$, and the bigger $||\mathbf{z}||$ is, the easier becomes solving SIS by finding such vector $\mathbf{v}$.

Thus, the norm of $||\mathbf{z}||$ is very important for the hardness of the ℓ_2-SIS$_{q,n,m,\beta}$ and that's why we want to keep $||\mathbf{z}||$ small. This situation is represented by the blue circle (2) because the problem here is not to find $\mathbf{S}$, but $\mathbf{v}$, where $\mathbf{v}$ depends on $||\mathbf{z}||$ that is about $\sqrt{m}||\mathbf{S}||$[7].

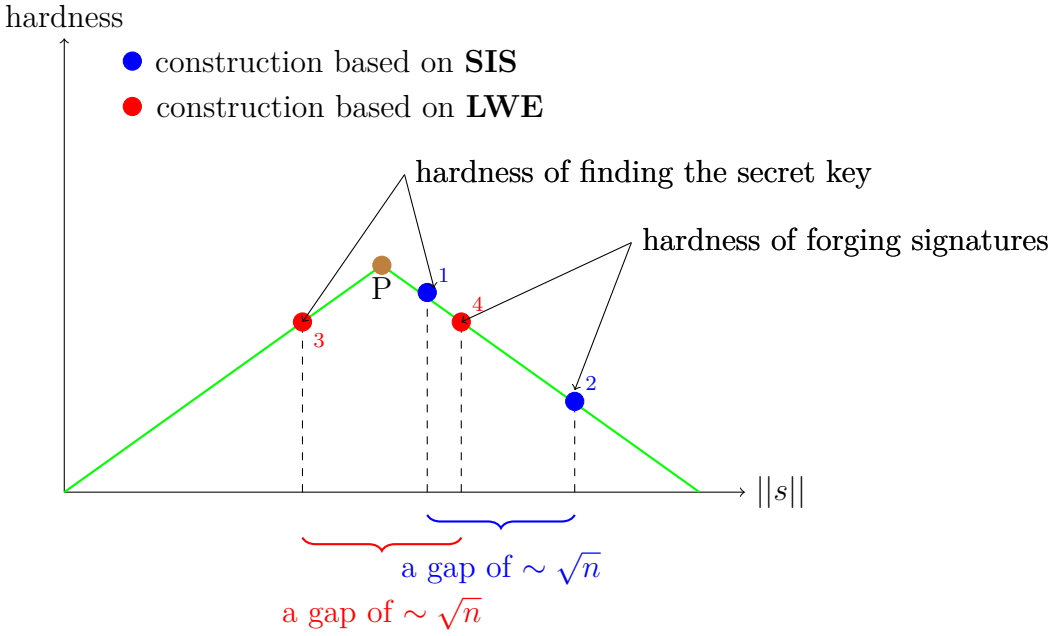


Figure 3.16: Hardness of finding the secret key and forging signature under SIS and LWE assumptions.

We can see how we have moved away from the point P, and the hardness of the

[7]From the section 3.5 we know that $||\mathbf{z}|| \approx \sigma\sqrt{m} = \alpha \cdot d \cdot \kappa \cdot m$, while $||\mathbf{Sc}|| \approx \kappa \cdot d \cdot \sqrt{m}$

problem of finding the secret key and the problem of forging signatures are not at the same level. Lyubashevsky's idea is to base the hardness of finding the secret key on the LWE problem so that the hardness of forging signatures will be at the same level around P. This situations is represented by the red circles (3) and (4).

Then, instead of requiring that given the public key the secret key must not be unique, we require, under the LWE assumption, that given the public key it's computationally indistinguishable whether the secret key is unique.

3.6.4 Ring variants of the signature scheme.

We just mention that the SIS and LWE problems can be extended to their Ring versions, resulting in key sizes k times smaller. The size of the signature is not influenced.

Chapter 4

The Bimodal Lattice Signature Scheme

In this chapter we will show how Lyubashevsky enhances the lattice-based digital signature scheme of Figure 3.1 by choosing as proposal distribution for the rejection sampling algorithm the Bimodal Gaussian distribution. We will see that on one hand the signature's size will be smaller, but on the other hand we will need to slightly modify the signing and the verification algorithms, and adopt a new key generation algorithm.

The signature scheme that exploits the Bimodal Gaussian in the rejection sampling algorithm has been presented by Lyubashevsky in his work "Lattice Signatures and Bimodal Gaussians" [11]. In the talks from 2013 [5] and 2016 [6] Lyubashevsky also spend some time explaining the bimodal optimization idea and the necessary modifications to the signing algorithm and verification algorithm, and for this reason in Appendix C we include the timestamps about these topics.

4.1 The trade-off of the shifted Gaussian distribution

Let us first recall that the parameter M represent the expected number of time the rejection sampling algorithm need to be repeated to produce an accepted sample, so we want M to be small, otherwise it would slow down the signing algorithm. In order to keep M small we must choose as proposal distribution g a distribution that quite resembles the target distribution f. Intuitively, we want to reduce the area between the scaled proposal distribution $M \cdot g$ and the target distribution f in order to reduce the number of rejected samples.

In the previous chapter, the proposal distribution g is the shifted Gaussian distribution while the target distribution f is the Gaussian distribution. The issue, in that choice for the proposal distribution, is that there is a tradeoff between the speed of rejection sampling algorithm and the length of the signature.

We can't really find a small value M such that $M \geq \dfrac{f(x)}{g(x)}$, $\forall x$, but Lemma 3.9 gives us a bound for M. Indeed, it tells us that if we set $\sigma = \alpha T$ (where $T = max||\mathbf{Sc}||$), then $\dfrac{f(x)}{g(x)} < e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}}$ with probability $1 - 2^{-100}$. This is the reason why we set $M = e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}}$ and $\sigma = \alpha T$. This is often addressed by Lyubahshevsky in his talks as the "problem at the tails", that is we can find a value for M that works only for all but the negligible

values of x[1].

Now consider, as example, the situation in Figure 4.1: there are the 1-dimensional target Gaussian distribution (in red) and the 1-dimensional shifted Gaussian distribution (in black) centered in $-v = -10$, and $\sigma = 2 \cdot ||v|| = 20$.

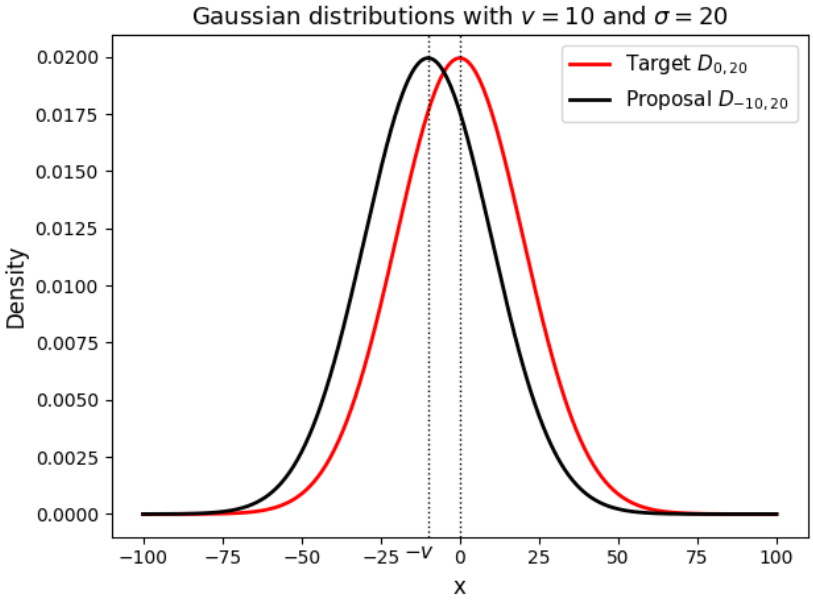


Figure 4.1: The proposal shifted Gaussian distribution $N(-10,20^2)$ and the target Gaussian distribution $N(0,20^2)$.

Suppose that we set $M = e \approx 2.72$. The appropriate value for σ would be $\sigma = 12 \cdot ||v|| = 120$, but we have set it to $\sigma = 2 \cdot ||v|| = 20$. Then, we expect that for some non negligible value x the scaled proposal shifted Gaussian doesn't envelope the target Gaussian because we have not respected the relation between M and σ given by Lemma 3.9. We can have a look at it in Figure 4.2. In Part 4.2a there is the plot of scaled proposal shifted Gaussian distribution along with the plot of the target Gaussian distribution, while in Part 4.2b there is a "zoom-in" on a portion of the graph where the proposal distribution does not envelope the target one.

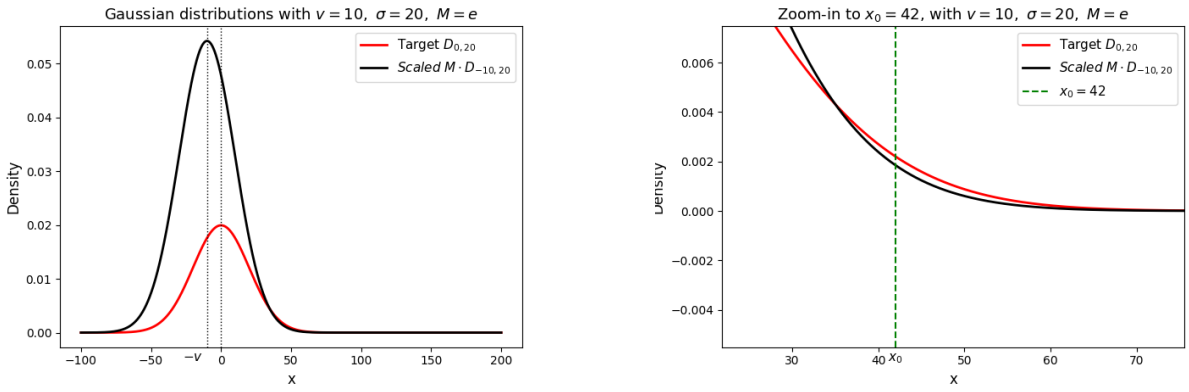
In Figure 4.3 we increase $\sigma = 4 \cdot ||v|| = 40$, but again when we consider the scaled proposal shifted Gaussian there will be a portion of the graph where the proposal distribution does not envelope the target one, as shown In Figure 4.4. Of course, the closer σ gets to the correct value ($\sigma = 12||v||$), the further from the mean $x = 0$ we will find some value x_0 where the scaled proposal shifted Gaussian distribution does not envelope the target Gaussian distribution, and for $\sigma = 12||v||$ those values will be the negligible values at the (right) tail of the Gaussian distribution.[2].

Then we must set M and σ accordingly to Lemma 3.9 and deal with the trade-off. Figure 4.5 shows the scaled proposal shifted Gaussian and the target Gaussian where $M = e$ and $\sigma = 12 \cdot ||v|| = 120$ is correctly set.

So, the trade-off given by Lemma 3.9 is that on one hand if α increases, then σ and the

[1]Negligible in the sense that they occur with a negligible probability of 2^{-100}.

[2]Since we consider a left shift we should look at the right tail, viceversa if we wanted to consider a right shift we would look at the left tail.



(a) The scaled proposal distribution **(b)** Zoom in

Figure 4.2: The scaled proposal and the target distribution for $M = e$ and $\sigma = 20$.

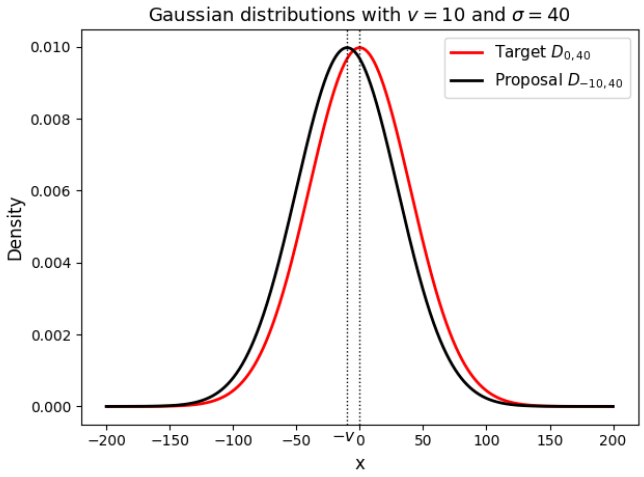
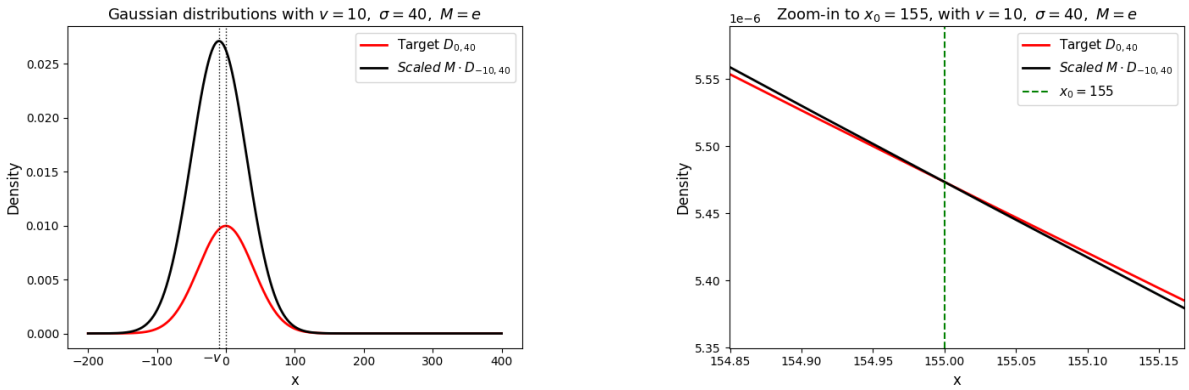


Figure 4.3: The proposal shifted Gaussian distribution $N(-10,40^2)$ and the target Gaussian distribution $N(0,40^2)$.



(a) The scaled proposal distribution **(b)** Zoom in

Figure 4.4: The scaled proposal and the target distribution for $M = e$ and $\sigma = 40$.

signature size increase, but M decreases which means the rejection sampling step is faster. On the other hand lowering α would slow down the rejection sampling step as M increase,

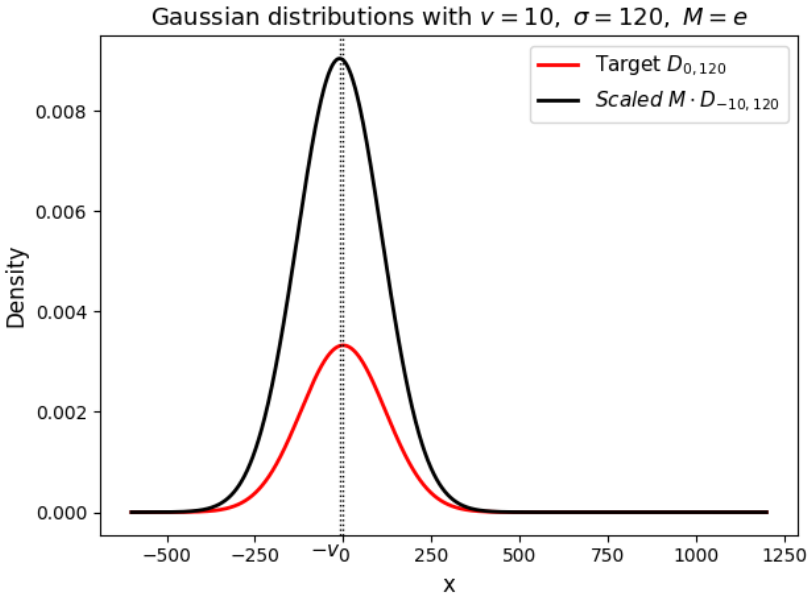


Figure 4.5: The scaled proposal shifted Gaussian distribution $N(-10,120^2)$ and the target Gaussian distribution $N(0,120^2)$.

but in turn σ and the signature size would be smaller. There is one more observation that we must point out: the maximum ratio between the scaled proposal shifted Gaussian and the target Gaussian is at the mean $x = 0$ and its surroundings. Then, we should expect that the most likely samples are also the ones most likely to be accepted after M iterations of the rejection sampling algorithm.[3]

We will see in the next section that the bimodal approach allow us not only to remove the "tail-cut" parameter α from the definition of σ, but also to reduce the ratio at the mean $x = 0$.

4.2 The bimodal gaussian idea and optimization

Let us recall that in the digital signature scheme of Figure 3.1 the response $\mathbf{z}$ is computed as $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$, and its distribution is the shifted Gaussian $D^m_{\mathbf{Sc},\sigma}$. Now suppose to compute the response $\mathbf{z}$ as $\mathbf{z} = b \cdot \mathbf{Sc} + \mathbf{y}$, where $b \xleftarrow{\$} \{-1,1\}$. Then, $\mathbf{z}$ is distributed according to the bimodal Gaussian $\frac{1}{2}D^m_{\mathbf{Sc},\sigma} + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}$.

So if the target distribution f is the Gaussian distribution D^m_σ, then

$$\frac{f(x)}{g(x)} = \frac{D^m_\sigma(x)}{\frac{1}{2}D^m_{\mathbf{Sc},\sigma}(x) + \frac{1}{2}D^m_{-\mathbf{Sc},\sigma}(x)} = \exp\left(\frac{||\mathbf{Sc}||^2}{2\sigma^2}\right) / \cosh\left(\frac{\langle \mathbf{x}, \mathbf{Sc}\rangle}{\sigma^2}\right) \leq \exp\left(\frac{||\mathbf{Sc}||^2}{2\sigma^2}\right)$$

So, if we set $\sigma = \alpha T$, where $T = max(||\mathbf{Sc}||)$ and $M = \exp\left(\frac{||\mathbf{Sc}||^2}{2\sigma^2}\right) = \exp\left(\frac{1}{2\alpha^2}\right)$, then if we want to obtain $M = e$ we just need $\alpha = 1/\sqrt{2}$ rather than $\alpha = 12$. Thus, $\sigma = T/\sqrt{2}$,

[3]It's common knowledge that for a Gaussian random variable about 68% of the probability mass lies within 1σ from the mean.

and comparing it to $\sigma = 12 \cdot T$ of 3.10, this is a reduction of about 94% of the value of the std.[4]

Let us make an example. Consider the situation in Figure 4.6: there are the 1-dimensional target Gaussian distribution and the 1-dimensional bimodal Gaussian distribution with modes in $-v$ and v, and $\sigma = ||v||\sqrt{2} = 10\sqrt{2} \approx 7.07$.

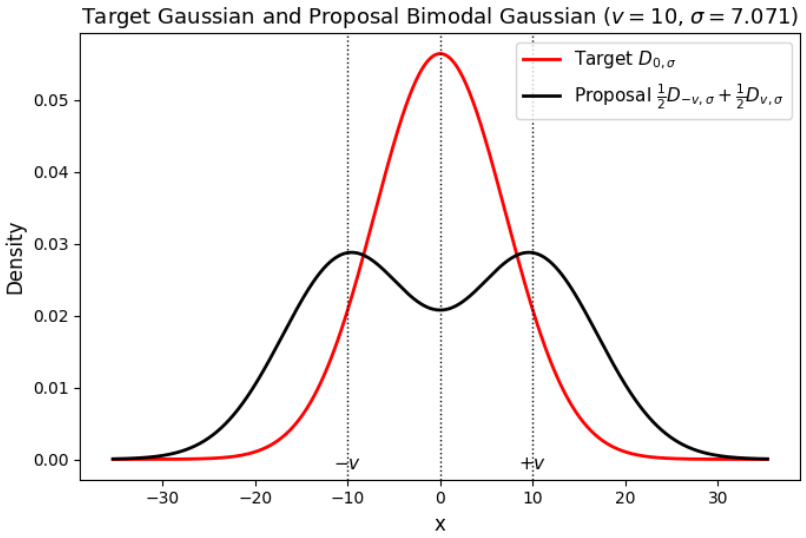


Figure 4.6: The proposal bimodal Gaussian distribution and the target Gaussian distribution.

When we scale the proposal bimodal Gaussian distribution by a factor $M = e$ we obtain the plot of Figure 4.7.

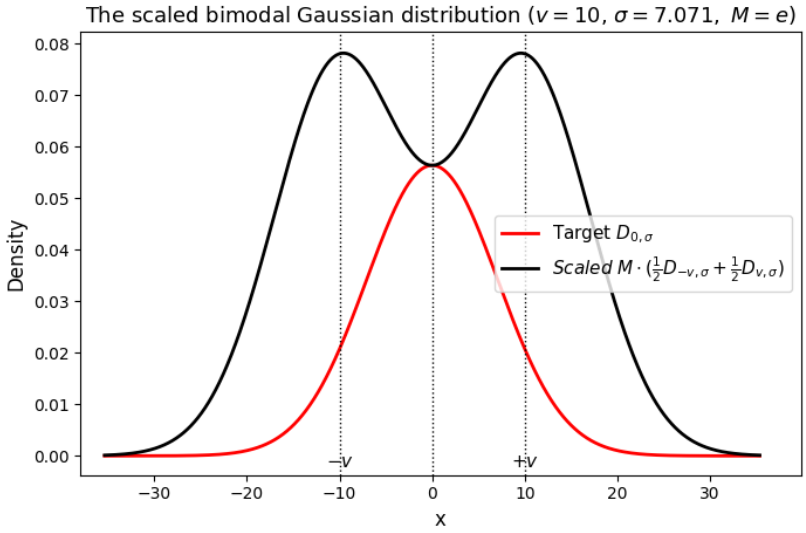


Figure 4.7: The scaled proposal bimodal Gaussian distribution and the target Gaussian distribution.

We can see that not only the scaled bimodal Gaussian envelopes the target distribution

[4]For a fixed value of T.

$\forall x$, but the proposal distribution presents two modes at $-v$ and $+v$. Then, the biggest ratio between the scaled proposal distribution and the target distribution is not at $x = 0$, but at $-v$ and $+v$. Indeed, the ratio $M \cdot g(x)/f(x)$ around $x = 0$ is quite small, meaning that we should expect the most likely samples to be accepted within a number of iterations of the rejection sampling algorithm less than M.

4.3 The BLISS digital signature scheme

In this section we will present two versions of the digital signature scheme called BLISS (Bimodal Lattice Signature Scheme), one based on the hardness of the SIS problem and one based on the hardness of the Ring-SIS problem. For this reason we provide the following definition of generalized SIS problem taken from [11, Section 2].

Definition 4.1 ($\mathcal{R}$-SIS$^{\mathcal{K}}_{q,n,m,\beta}$ problem)**.** *Let $\mathcal{R}$ be some ring and $\mathcal{K}$ be some distribution over $\mathcal{R}_q^{n\times m}$, where $\mathcal{R}_q$ is the quotient ring $\mathcal{R}/(q\mathcal{R})$. Given a random $\mathbf{A} \in \mathcal{R}_q^{n\times m}$ drawn according to the distribution $\mathcal{K}$, find a non-zero $\mathbf{v} \in \mathcal{R}_q^m$, such that $\mathbf{Av} = \mathbf{0}$ and $||\mathbf{v}||_2 \leq \beta$.*

We will also discuss the parameters involved with both versions of the signatures schemes, highlighting the differences with the digital signature scheme from Figure 3.1.

4.3.1 Basing BLISS on the classic SIS problem

If we consider $\mathcal{R} = \mathbb{Z}$ and $\mathcal{K}$ to be the uniform distribution, then definition 4.1 is exactly the definition of the classical SIS problem.

Signature Algorithm

Input: Message μ, public key $\mathbf{A} \in \mathbb{Z}_{2q}^{n\times m}$, secret key $\mathbf{S} \in \mathbb{Z}_{2q}^{m\times n}$, std $\sigma \in \mathbb{R}$
Output: A signature $(\mathbf{z}, \mathbf{c})$ of the message μ

1: $\mathbf{y} \xleftarrow{\$} D_\sigma^m$
2: $\mathbf{c} \leftarrow H(\mathbf{Ay} \bmod 2q, \mu)$
3: Choose a random bit $b \in \{0,1\}$
4: $\mathbf{z} := (-1)^b\mathbf{Sc} + \mathbf{y}$
5: **Output** $(\mathbf{z}, \mathbf{c})$ with probability

$$\frac{1}{M \cdot \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cdot \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc}\rangle}{\sigma^2}\right)},$$

otherwise **restart**.

Figure 4.8: Signature Algorithm

By looking at Figure 4.8 we can see that the signature scheme [11, Section 3.1] is quite different from the one in figure 3.1.

- The public key $\mathbf{A}$ is a $n \times m$ matrix whose coefficients are uniformly random in $\mathbb{Z}_{2q}$.
- The secret key $\mathbf{S}$ is a (short) $m \times n$ matrix in $\mathbb{Z}_{2q}$.

- The response $\mathbf{z}$, as we have already presented, is computed by picking a random bit b and computing $\mathbf{z} := (-1)^b\mathbf{Sc} + \mathbf{y}$.
- Finally, the probability at step 5) is a direct consequence of the rejection sampling step using as proposal distribution the Bimodal Gaussian discussed in the previous section.

We mention that the hash function H is modeled as a random oracle whose output domain is the set $\mathbb{B}_\kappa^n$ of binary vectors of length n and weight κ.[5] The details on how the key-pair is generated will be addressed prior the discussion of the modification of the verification algorithm reported in Figure 4.9 [11, Section 3.1].

Verification Algorithm

Input: Message μ, public key $\mathbf{A} \in \mathbb{Z}_{2q}^n$, signature $(\mathbf{z}, \mathbf{c})$
Output: Accept or Reject the signature
1: **if** $\|\mathbf{z}\| > B_2$ **then** Reject
2: **if** $\|\mathbf{z}\|_\infty \geq q/4$ **then** Reject
3: Accept iff $\mathbf{c} = H(\mathbf{Az} + q\mathbf{c} \bmod 2q, \mu)$

Figure 4.9: Verification Algorithm.

The BLISS verification algorithm checks that the norm of the signature $||\mathbf{z}||$ is lower than some bound $B_2 = \eta \cdot \sqrt{m} \cdot \sigma$, where η is set so that $||\mathbf{z}|| \leq B_2$ with probability $1 - 2^{-100}$ (see Lemma 3.8). The additional check $||\mathbf{z}||_\infty \leq q/4$ comes from the security proof described in [11] and it's usually verified whenever the first one is.

The verification algorithm of the digital signature in Figure 3.1 checks that $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$, but in BLISS it doesn't work anymore because it's no longer guaranteed that $\mathbf{Ay} = \mathbf{Az} - \mathbf{Tc} \bmod q$. Indeed, $\mathbf{Ay} = \mathbf{Az} - \mathbf{Tc} = \mathbf{A}(b\mathbf{Sc} + \mathbf{y}) - \mathbf{Tc} = \mathbf{Ay} + b\mathbf{Tc} - \mathbf{Tc}$, which is true if $b\mathbf{Tc} = \mathbf{Tc} \bmod q$ for $b \in \{-1,1\}$. Now, as q is a prime number, $\mathbf{Tc} = -\mathbf{Tc} \bmod q$ it's impossible unless $\mathbf{T} = \mathbf{0}$. Then, the solution devised by Lyubashevsky is to work modulo $2q$ and set $\mathbf{T} = q\mathbf{I}$, where $\mathbf{I}$ is the $n \times n$ identity matrix.

So, the verification procedure is fixed, but since the matrix $\mathbf{T}$ is no longer computed as $\mathbf{A} \cdot \mathbf{S} \bmod q$, the key generation procedure must be reviewed too.

A new key generation algorithm. We now describe how to generate the keys $\mathbf{A}, \mathbf{S}$ such that $\mathbf{AS} = \mathbf{T} = q\mathbf{I} \pmod{2q}$ [11, Appendix B].

Let's start by defining $m = m' + n$ and choosing an uniform matrix $\mathbf{A'}_q \in \mathbb{Z}_q^{n \times m'}$ and a random small matrix $\mathbf{S'} \in \mathbb{Z}_q^{m' \times n}$ with coefficients in $(-2^\alpha, 2^\alpha)$. Consider the matrix $\mathbf{A}_q = (\mathbf{A'}_q \mid -\mathbf{A'}_q\mathbf{S'}) \in \mathbb{Z}_q^{n \times m}$. One can show that the statistical distance between the distribution of $\mathbf{A}_q$ and the uniform distribution over $\mathbb{Z}_q^{n \times m}$ is at most $n \cdot 1/2\sqrt{q^n/2^{(\alpha+1)} \cdot m'}$. Thus, we need to set

$$m' \geq \frac{2(\lambda - 1 + \lceil \log_2 n \rceil) + n\lceil \log_2 q \rceil}{\alpha + 1}$$

[5] κ represent the number of non-zero coefficients.

in order for this statistical distance to be negligible in the security parameter λ.

The final step is to set the secret key $\mathbf{S} = (\mathbf{S}', \mathbf{I}_n)^T \in \mathbb{Z}_{2q}^{m \times n}$ and the public key $\mathbf{A} = (2\mathbf{A'}_q \mid q\mathbf{I}_n - 2\mathbf{A'}_q\mathbf{S}') \in \mathbb{Z}_{2q}^{n \times m}$. The one easily checks that $\mathbf{AS} = q\mathbf{I}_n$.

Analysis of the parameters

We now analyze the parameters of the BLISS digital signature scheme based on the SIS problem. We must say that the parameters we will discuss are not present in Lyubashevsky's paper, but are proposed by us. Our goal is to consider column II of table 3.1 and modify the parameters as if the bimodal gaussian idea is employed in the rejection sampling step of the signature scheme in 3.1. As the choice of adopting the bimodal gaussian mainly impacts the parameters M and α, we just want to show how those parameters change while the others remain almost unchanged.

Parameter	I	II
λ	100	100
n	512	512
q	2^{25}	2^{25}
κ	14	14
α	0.5	$\frac{1}{\sqrt{2}}$
$M \approx \exp(\frac{1}{2\alpha^2})$	7.4	2.72
$d = \lfloor 2^\alpha \rfloor$	1	1
$m' \approx \frac{2(\lambda - 1 + \lceil \log_2 n \rceil) + n\lceil \log_2 q \rceil}{\alpha + 1}$	8677	7625
$m = m' + n$	9189	8137
$\sigma \approx \alpha \cdot d \cdot \kappa \cdot \sqrt{m}$	671	893
approx. signature size (bits) $\approx m \log(12\sigma)$	119000	108900
approx. secret key size (bits) $\approx m \cdot n \cdot \log(2d+1)$	$2^{22.8}$	$2^{22.5}$
approx. public key T size (bits) $\approx n \cdot n \cdot \log q$	$2^{22.5}$	$2^{22.5}$

Table 4.1: Parameters of the BLISS scheme based on the SIS problem

The parameters λ, n, q, κ as well as the formulas for computing the sizes of the signature and the keys[6] are left unchanged from column II of table 3.1.

Since we are adopting the bimodal gaussian as proposal distribution for the rejection sampling step, on one hand the bound on M can be set equal to $\exp(\frac{1}{2\alpha^2})$ as we have seen in section 4.2, while the std $\sigma = \alpha T$ is approximately $\alpha \cdot d \cdot \kappa \cdot \sqrt{m}$ as we have seen in section 3.5. Then, by setting the value of α we can tune M and σ.

[6]The parameters k has been substituted by n

In column I of table 4.1 we set $\alpha = 0.5$, while in column II we set $\alpha \approx 0.707$. We can see that the small variation of the parameter has a significant effect on M that decrease from 7.4 to 2.72, and σ, which increases from 671 up to 893.

The comparison between the values in column II of table 4.1 with the ones of column II of table 3.1 already showcase the power of the bimodal gaussian approach.

- the std σ decreased from 15157 to 893, that is a reduction of about 94%, as we expected
- the size of the signature decreased from 142300 to 108900 as a consequence of the reduction of σ

The sizes of the secret and public keys did not change.

4.3.2 Basing BLISS on the Ring-SIS problem

We are now considering $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ so that definition 4.1 is the definition of the Ring-SIS problem.

Since the keys must live in the ring $\mathcal{R}_q$, Lyubashevsky's new key generation algorithm was inspired by the NTRU key generation method [11, Section 4]. We'll report below the procedure to generate the secret key $\mathbf{S}$ and the public key $\mathbf{A}$ such that $\mathbf{AS} = q\mathbf{I}_q = \mathbf{T}$.

Key generation

First of all we need two densities $\delta_1, \delta_2 \in [0,1]$ and generate two random polynomials $\mathbf{f}, \mathbf{g}$ with exactly $d_1 = \lceil \delta_1 n \rceil$ coefficients in $\{\pm 1\}$ and exactly $d_2 = \lceil \delta_2 n \rceil$ coefficients in $\{\pm 2\}$, while the other $(n - d_1 - d_2)$ coefficients are set to 0. The generation of $\mathbf{f}$ shall be repeated until it is invertible. Then we can set $\mathbf{S} = (s_1, s_2)^T = (\mathbf{f}, 2\mathbf{g} + 1)^T$.

The public key can be set as $\mathbf{A} = (2\mathbf{a}_q, q-2) \in \mathcal{R}_{2q}^{1\times 2}$, where $\mathbf{a}_q = s_2/s_1 = (2\mathbf{g}+1)/\mathbf{f} \in \mathcal{R}_q$ is defined as a quotient modulo q. Then one can easily check that $\mathbf{AS} = q \bmod 2q$.

A new bound on the norm of Sc

When computing $\sigma = \alpha T$, where $T = max(||\mathbf{Sc}||)$, we considered as bound on the norm of $\mathbf{Sc}$ the quantity $d \cdot \kappa \cdot \sqrt{m}$, but we can do better. Lyubashevsky defines a "new measure of $\mathbf{S}$ adapted to the form of $\mathbf{c}$" [11, Section 3] which helps achieving a tighter bound on $||\mathbf{Sc}||$.

Definition 4.2. *For any integer κ, we define $N_\kappa : \mathbb{R}^{m\times n} \to \mathbb{R}$ as:*

$$N_k(\mathbf{X}) = \max_{I\subset\{1,\dots,n\},\#I=\kappa} \sum_{i\in I} \left(\max_{J\subset\{1,\dots,n\},\#J=\kappa} \sum_{j\in J} T_{i,j} \right) \quad where \quad \mathbf{T} = \mathbf{X}^T \cdot \mathbf{X} \in \mathbb{R}^{n\times n}.$$

Proposition 4.3. *Let $\mathbf{S} \in \mathbb{R}^{m\times n}$ be a real matrix. For any $\mathbf{c} \in \mathbb{B}_\kappa^n$, we have $||\mathbf{Sc}||^2 \leq N_\kappa(\mathbf{S})$.*

The last proposition tells us that $\sqrt{N_\kappa(\mathbf{S})}$ is an upper bound for $||\mathbf{Sc}||$.

The measure $N_\kappa(\mathbf{S})$ is actually also very important for the key generation step. After the computation of the secret key $\mathbf{S}$, we discard it if $N_\kappa(\mathbf{S}) \geq C^2 \cdot 5 \cdot (d_1 + 4d_2) \cdot \kappa$ for a fixed constant C.

We try to explain the motivation behind this rejection step in the key generation process.

In our previous digital signature schemes the bound for the norm of $\mathbf{Sc}$ was given by the quantity $d \cdot \kappa \cdot \sqrt{m}$, which is a worst-case bound valid for all the secret keys $\mathbf{S} \in \{-d, ..., d\}^{m \times n}$ when $\mathbf{c}$ is a sparse ternary (or binary) vector with exactly κ non-zero entries. This worst-case bound depends only on d, κ, m and not on the specific value of $\mathbf{S}$. Thus, the acceptance probability of a signature depends only on this bound, which is the same for all possible keys.

On the contrary, when we adopt this new bound $\sqrt{N_\kappa(\mathbf{S})}$, the quantity $N_\kappa(\mathbf{S})$ depends on the actual secret key $\mathbf{S}$, and the acceptance probability would vary with the secret key too. This means that if two different secret keys $\mathbf{S}, \mathbf{S}'$ produce different $N_\kappa(\mathbf{S}), N_\kappa(\mathbf{S}')$, the observed acceptance rates for those keys will differ, and possibly an attacker who sees many signatures (or measure signing latencies) can infer something about the secret key being used. Thus, if we limit this bound to be a fixed quantity $C^2 \cdot 5 \cdot (d_1 + 4d_2) \cdot \kappa$ it will be valid for every accepted secret key. We will call this bound $\sqrt{N_\kappa}$.

The constant C is chosen so that «only 25% of the keys are accepted, decreasing the overall security by at most 2 bits» [11, Section 4.1]. Indeed, because the space of accepted keys is $\frac{1}{4}$ of the original one, an attacker's brute force search space is 4 times smaller. Then, we lose 2 bits of security against brute force attacks, but that's not really an issue if we work with sufficiently high values of the security parameter λ, like 128,160,192.

In figure 4.10 we report the key generation algorithm of the BLISS digital signature [11, Section 4.8].

BLISS Key Generation Algorithm

Output: Key pair $(\mathbf{A}, \mathbf{S})$ such that $\mathbf{AS} = q \bmod 2q$

1: Choose $\mathbf{f}, \mathbf{g}$ as uniform polynomials with exactly d_1 entries in $\{\pm 1\}$ and d_2 entries in $\{\pm 2\}$
2: $\mathbf{S}{=}(\mathbf{s_1}, \mathbf{s_2})^T \leftarrow (\mathbf{f}, 2\mathbf{g} + 1)^T$
3: **if** $N_\kappa(\mathbf{S}) \geq C^2 \cdot 5 \cdot (\lceil \delta_1 n \rceil + 4\lceil \delta_2 n \rceil) \cdot k$ **then**
4: **restart**
5: **end if**
6: $\mathbf{a}_q = (2\mathbf{g} + 1)/\mathbf{f} \bmod q$ (**restart** if $\mathbf{f}$ is not invertible)
7: **Output** $(\mathbf{A}, \mathbf{S})$ where $\mathbf{A} = (\mathbf{a_1}, q - 2) \bmod 2q = (2\mathbf{a}_q, q - 2) \bmod 2q$

Figure 4.10: BLISS Key Generation Algorithm

The signing and verification algorithms

We now present and briefly comment the signing and verification scheme of the BLISS digital signature [11, Section 4.8].

The main difference with the previous digital signatures schemes is that the public key consist of two elements in $\mathcal{R}_q$, so we want to produce, in the first step, $\mathbf{y} = (\mathbf{y_1}, \mathbf{y_2})^T$, where $\mathbf{y_1}, \mathbf{y_2}$ are two polynomials with coefficients distributed according to a centered discrete Gaussian distribution with std σ. Then, we give $\mathbf{Ay} \bmod 2q$ and μ as input to the random oracle H just like we did in 4.8 to obtain a challenge $\mathbf{c}$. Finally, we pick a random bit b, compute $\mathbf{z_1} = \mathbf{y_1} + (-1)^b \mathbf{s_1 c}$ and $\mathbf{z_2} = \mathbf{y_2} + (-1)^b \mathbf{s_2 c}$ and perform rejection

sampling to accept/reject the pair $(\mathbf{z_1}, \mathbf{z_2})$. If the pair $(\mathbf{z_1}, \mathbf{z_2})$ is accepted we output the signature pair $(\mathbf{z}, \mathbf{c})$ where $\mathbf{z} = (\mathbf{z_1}, \mathbf{z_2})^T$ follows the Gaussian distribution D_σ^{2n}.

In the BLISS signing algorithm provided in Figure 4.11 Lyubashevsky actually adopts a signature compression optimization which allows to shrink even more the signature size, first by dropping the low order bits of $\mathbf{z_2}$ and then by employing Huffman encoding [11, Section 4.7] to compress the highest bits of $\mathbf{z_1}, \mathbf{z_2}$. To carry out this optimization the signing algorithm present two additional steps.

In the step 2 we compute the first input $\mathbf{u} = \zeta \cdot \mathbf{a_1} \cdot \mathbf{y_1} + \mathbf{y_2} \bmod 2q$ of the random oracle H. The parameter ζ is chosen so that $\zeta\mathbf{A} = (\zeta\mathbf{a_1}, 1)$, that is $\zeta \cdot (q-2) = 1 \bmod 2q$.

In the step 8 we let $\mathbf{z_2^\dagger} \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$ be the value of $\mathbf{z_2}$ with its low-order bits truncated. The parameter d represent the number of dropped bits, while the parameter p is defined as $p = \lfloor 2q/2^d \rceil$.

The signature is the triple $(\mathbf{z_1}, \mathbf{z_2^\dagger}, \mathbf{c})$. We won't go into the details of the optimization procedure.

BLISS Signing Algorithm

Input: Message μ, public key $\mathbf{A} = (\mathbf{a}_1, q-2) \in R_{2q}^{1\times 2}$, secret key $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^T \in R_{2q}^{2\times 1}$
Output: A signature $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$ of the message μ

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_{\mathbb{Z}^n,\sigma}$
2: $\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$
3: $\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$
4: Choose a random bit b
5: $\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
6: $\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
7: **Continue** with probability

$$1 \Big/ \left(M \cdot \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2} \right) \cdot \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2} \right) \right),$$

otherwise **restart**.
8: $\mathbf{z_2^\dagger} \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
9: **Output** $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$

Figure 4.11: BLISS Signing Algorithm

BLISS Verification Algorithm

Input: Message μ, public key $\mathbf{A} = (\mathbf{a}_1, q-2) \in R_{2q}^{1\times 2}$, signature $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
Output: Accept or Reject the signature

1: **if** $\|\mathbf{z}_1 \mid 2^d \cdot \mathbf{z}_2^\dagger\|_2 > B_2$ **then** Reject
2: **if** $\|\mathbf{z}_1 \mid 2^d \cdot \mathbf{z}_2^\dagger\|_\infty > B_\infty$ **then** Reject
3: Accept iff $\mathbf{c} = H(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z_1} + \zeta \cdot q \cdot \mathbf{c} \rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu)$

Figure 4.12: BLISS Verification Algorithm

The verification algorithm in Figure 4.12 is modified accordingly to check that the l_2-norm of $\mathbf{z} = (\mathbf{z_1}, \mathbf{z_2^\dagger})$ is lower than some bound B_2 and the infinity norm of $\mathbf{z} = (\mathbf{z_1}, \mathbf{z_2^\dagger})$ is lower than some bound B_∞. Finally, we check that $\mathbf{c} = H(\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z_1} + \zeta \cdot q \cdot \mathbf{c} \rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu)$.

Analysis of the parameters

We report in Table 4.2 the parameters proposed by Lyubashevsky for the BLISS digital signature scheme [11, Section 5].

Name of the scheme	**BLISS-0**	**BLISS-I**	**BLISS-II**	**BLISS-III**	**BLISS-IV**
Security parameter λ	≤ 60 bits	128 bits	128 bits	160 bits	192 bits
Optimized for	Fun	Speed	Size	Security	Security
n	256	512	512	512	512
q	7681	12289	12289	12289	12289
Secret key densities δ_1, δ_2	.55, .15	.3, 0	.3, 0	.42, .03	.45, .06
Gaussian standard deviation $\sigma = \alpha \cdot \sqrt{N_\kappa}$	100	215	107	250	271
α	.5	1	.5	.7	.55
$M \approx \exp(\frac{1}{2\alpha^2})$	7.4	1.6	7.4	2.8	5.2
κ	12	23	23	30	39
Secret key N_κ-threshold C	1.5	1.62	1.62	1.75	1.88
Dropped bits d in $\mathbf{z_2}$	5	10	10	9	8
Verification thresholds (B_2, B_∞)	2492, 530	12872, 2100	11074, 1563	10206, 1760	9901, 1613
Signature size	**3.3 kb**	**5.6 kb**	**5 kb**	**6 kb**	**6.5 kb**
Secret key size	1.5 kb	2 kb	2 kb	3 kb	3 kb
Public key size	3.3 kb	7 kb	7 kb	7 kb	7 kb

Table 4.2: Parameters proposal for the BLISS digital signature

The parameters α, M, κ, C and σ. Once we set α, we will directly determine the value of M because $M \approx \exp(\frac{1}{2\alpha^2})$. The parameters κ and C are chosen by Lyubashevsky so that only 25% of the secret keys would satisfy the constraint $N_\kappa(\mathbf{S}) \leq C^2 \cdot 5 \cdot (d_1 + 4d_2) \cdot \kappa$.

The std σ can be easily computed as the product of α times the bound $\sqrt{N_\kappa}$, that is $\sigma = \alpha \cdot \sqrt{C^2 \cdot 5 \cdot (d_1 + 4d_2) \cdot \kappa}$.

The keys size. Let's recall that the secret key is $\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^T = (\mathbf{f}, 2\mathbf{g} + 1)^T \in R_{2q}^{2\times 1}$, where $\mathbf{f}, \mathbf{g} \in \mathcal{R}_q$ are polynomials with exactly $d_1 = \lceil \delta_1 n \rceil$ coefficients in $\{\pm 1\}$ and exactly $d_2 = \lceil \delta_2 n \rceil$ coefficients in $\{\pm 2\}$, while the public key is $\mathbf{A} = (2\mathbf{a}_q, q - 2) \in R_{2q}^{1\times 2}$ where $\mathbf{a}_q = s_2/s_1 = (2\mathbf{g} + 1)/\mathbf{f} \in \mathcal{R}_q$.

Then, the secret key is composed of two polynomials in which each of the n coefficients can have a value in $\{-2, -1,\ 0, +1, +2\}$[7]. For this reason we can determine the secret key size as $2 \cdot n \cdot \lceil \log_2 5 \rceil$ (bits). Please note that if $\delta_2 = 0$ that's not true anymore and we shall determine the secret key size as $2 \cdot n \cdot \lceil \log_2 3 \rceil$ (bits) because the polynomials $\mathbf{f}, \mathbf{g}$ will be ternary polynomials (with coefficients in $\{-1,\ 0, +1\}$).

In a similar way we can compute the size of the public key. Because we can't know a priori the values of the coefficients of $\mathbf{a}_q$, we will assume each coefficient takes a value in Z_q. Therefore the public key size is given by approximately[8] $n \cdot \lceil \log_2 q \rceil$ (bits).

The signature size. As previously discussed, Lyubashevsky adopts several techniques to reduce the signature size through compression. Table 4.2 reports the sizes obtained with these optimizations. However, we would like to present here the signature sizes without compression to highlight that they remain relatively short even in the absence of such optimizations.

To do this, we compute the size of the signature $\mathbf{z} = (\mathbf{z}_2, \mathbf{z}_2)$ with the usual formula $[n \cdot \log_2(12\sigma)] \cdot 2$. We obtain the following results:

- **BLISS-0**: 5.2 kb
- **BLISS-I**: 11.6 kb
- **BLISS-II**: 10.5 kb
- **BLISS-III**: 11.8 kb
- **BLISS-IV**: 11.9 kb

A performance comparison with RSA and ECDSA digital signatures

Table 4.3 summarizes the performance of the BLISS signature scheme compared with the very well known classical digital signatures of RSA and ECDSA. The reported values are from a proof-of-concept implementation on standard 64-bit processor (desktop computer, Intel Core i7 at 3.4Ghz, 32GB RAM), and BLISS already shows competitive results despite the absence of heavy optimization. The performance results in table 4.3 are taken from [11, Section 1], while at [13] there's the implementation of the BLISS scheme which have been used for the benchmark.

A first observation is the cost of verification: for all BLISS parameter sets, verification takes about 0.03 ms, which is remarkably consistent across the different security levels.

[7] We don't know a priori the value of each coefficient, so we must represent all the coefficients with the same number of bits.

[8] We don't take into account the second term of the public key $(q - 2)$

This makes BLISS roughly one 10x faster than RSA verification and between 10× and 30× faster than ECDSA verification, depending on the key size.

In terms of signing, the picture is more balanced. BLISS-I achieves a signing time of 0.124 ms, essentially on par with ECDSA-256 (0.106 ms) that offers the same security level. The Other BLISS variants incur in a little higher signing costs, though they remain within the same order of magnitude as ECDSA and are considerably faster than high-security RSA signatures (e.g., RSA-2048 or RSA-4096). An additional consideration is that, unlike RSA and ECDSA where the signing cost grows proportionally with the targeted security level, the signing time in BLISS does not scale proportionally with the bit-security parameter. Instead, it is mainly influenced by the parameter M used in the rejection sampling step of the signing algorithm, which governs the number of expected attempts to produce a valid signature. As a consequence, two BLISS variants with the same security level (BLISS-I and BLISS-II) show noticeably different signing times: the signing time of BLISS-II is about 4 times larger than the one of BLISS-I, quite reflecting the ratio among the values of M ($M_{BLISS-I} = 1.6, M_{BLISS-II} = 7.4$).

Implementation	Security	Signature Size	SK Size	PK Size	Sign (ms)	Sign/s	Verify (ms)	Verify/s
BLISS-0	60 bits	3.3 kb	1.5 kb	3.3 kb	0.241	4k	0.017	59k
BLISS-I	128 bits	5.6 kb	2 kb	7 kb	0.124	8k	0.030	33k
BLISS-II	128 bits	5 kb	2 kb	7 kb	0.480	2k	0.030	33k
BLISS-III	160 bits	6 kb	3 kb	7 kb	0.203	5k	0.031	32k
BLISS-IV	192 bits	6.5 kb	3 kb	7 kb	0.375	2.5k	0.032	31k
RSA 1024	72-80 bits	1 kb	1 kb	1 kb	0.167	6k	0.004	91k
RSA 2048	103-112 bits	2 kb	2 kb	2 kb	1.180	0.8k	0.038	27k
RSA 4096	$\geq$ 128 bits	4 kb	4 kb	4 kb	8.660	0.1k	0.138	7.5k
ECDSA 160	80 bits	0.32 kb	0.16 kb	0.16 kb	0.058	17k	0.205	5k
ECDSA 256	128 bits	0.5 kb	0.25 kb	0.25 kb	0.106	9.5k	0.384	2.5k
ECDSA 384	192 bits	0.75 kb	0.37 kb	0.37 kb	0.195	5k	0.853	1k

Table 4.3: Comparison of BLISS, RSA, and ECDSA performance

Regarding key and signature sizes, BLISS signatures are larger than those of ECDSA (several kilobits versus less than a kilobits), but remain comparable to RSA at equivalent security levels. For instance, BLISS-I signatures ($\approx$ 5.6 kb) are roughly the same size as RSA-4096 signatures ($\approx$ 4 kb), both of which aim at 128-bit security. The private key sizes in BLISS (2–3 kb) are also not significantly larger than their RSA counterparts.

Overall, the comparison highlights two key takeaways. First, BLISS offers much faster verification than both RSA and ECDSA, which is a strong advantage in real-world deployments. Second, its signing performance is better than RSA and it's competitive with ECDSA at the same security level. The main trade-off lies in the signature size, which is larger than in classical schemes, but still within acceptable limits for most applications.

Chapter 5

Conclusion

In conclusion, this thesis work provided a detailed explanation of the rejection sampling techniques and its use in the context of lattice-based digital signatures, as a tool to achieve independence of the signature from the secret key. Then, with the analysis of the parameters used in the digital signature from "Lattice Signatures Without Trapdoors" we highlighted the trade-off between speed and signature size and how rejection sampling plays a crucial role in this trade.

We showed how with the SIS-based signature scheme and 100 bits of security we were able to obtain signatures of size of about 100 kb, while with the low-density SIS or LWE versions of the signature scheme we achieved signatures of size of 15-20 kb. However, we must take into account that the latter approach requires to find the right balance (i.e. achieve the same security level) between the low-density SIS and the SIS problems.

We then presented the BLISS signature scheme, that has a more straightforward way to obtain smaller signatures, that is the adoption of bimodal Gaussian in the rejection sampling step, without requiring any extra security assumptions beyond the SIS one. On the one hand, this change entailed the burden of modifying the key-generation approach, but on the other hand, it allowed us to obtain signatures of about 5-6 kb for 128 bits (and more) bits of security.

So, the BLISS signature scheme produces signatures whose sizes are much smaller than those of the signatures produced by the signature scheme of "Lattice Signatures Without Trapdoors". Also, the benchmark results showed that the BLISS scheme is competitive with the classical RSA and ECDSA signatures schemes, especially on signature verification speed.

Despite these promising results, the main challenge of using Gaussians in the rejection sampling step is the secure and efficient implementation of a Gaussian sampler[1], especially in constrained devices with limited memory and power [11].

Future thesis work

A possible continuation of this thesis project is to explore the techniques used by Lyubashevsky to obtain efficient lattice-based zero-knowledge proofs, in particular the ABDLOP commitment scheme and the Johnson-Lindenstrauss lemma . The underlying problem, strictly related to the lattice-based identification scheme — the one that is transformed into

[1]The Gaussian sampler is required to sample y in the first step of the signing algorithm

the digital signature schemes analyzed in this thesis work — developed by Lyubashevsky is illustrated in Appendix B.

Appendix A

Python Scripts

We will provide only the relevant python scripts used for the experiments whose results are shown in the chapters.

A.1 Scripts used for Chapter 3

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta, uniform

# Parameters
alpha = 2
beta_param = 4
a = 0
b = 1
M = 2.2  # scaling factor
x0 = 0.5  # sampled x value

# Generate 500 x values
x = np.linspace(0, 1, 500)

# PDF values
beta_pdf = beta.pdf(x, alpha, beta_param)
uniform_pdf = uniform.pdf(x, loc=a, scale=b-a) * M

# Evaluate at x0
f_x0 = beta.pdf(x0, alpha, beta_param)
Mg_x0 = uniform.pdf(x0, loc=a, scale=b-a) * M

# Plot Beta(2,4) in blue
plt.figure(figsize=(6, 4))
plt.plot(x, beta_pdf, color='blue', lw=2, label=f'Beta({alpha},{
    beta_param})')

# Plot scaled Uniform in orange
plt.plot(x, uniform_pdf, color='orange', lw=2, label=f'M*Uniform
    ({a},{b})')

```

```
# Draw vertical lines for the Uniform boundaries
plt.vlines(x=a, ymin=0, ymax=max(uniform_pdf), colors='orange',
    linestyles='--')
plt.vlines(x=b, ymin=0, ymax=max(uniform_pdf), colors='orange',
    linestyles='--')

# Mark the points f(x0) and Mg(x0)
plt.scatter(x0, f_x0, color='blue', zorder=5)
plt.scatter(x0, Mg_x0, color='orange', zorder=5)

# Add labels near points
plt.text(x0 + 0.02, f_x0, r"$f(x_0)$", color='blue', fontsize=10,
     va='center')
plt.text(x0 + 0.02, Mg_x0-0.08, r"$M g(x_0)$", color='orange',
    fontsize=10, va='center')

# Add green evaluation line from f(x0) to Mg(x0)
plt.vlines(x=x0, ymin=0, ymax=Mg_x0, colors='green', lw=2)

# Labels and styling
plt.title(f'Rejection Sampling: Beta({alpha},{beta_param}) and M*
    Uniform({a},{b})')
plt.xlabel('x')
plt.ylabel('Density')
plt.ylim(0, max(uniform_pdf) * 1.1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.show()
```

Listing A.1: script that generates Figure 3.5

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta, uniform

# Parameters
alpha = 2
beta_param = 4
a = 0
b = 1
M = 2.2
n_samples = 1000

# Proposal distribution g(x) = Uniform(a,b)
def g_pdf(x):
    return uniform.pdf(x, loc=a, scale=b-a)

# Target distribution f(x) = Beta(alpha, beta_param)
def f_pdf(x):
    return beta.pdf(x, alpha, beta_param)

# --- Rejection sampling  ---
# sample X ~ g, draw U ~ Unif(0,1), set Y = U * M * g(X)
x_proposals = np.random.uniform(a, b, size=n_samples)
```

```
u = np.random.uniform(0, 1, size=n_samples)
y_props = u * (M * g_pdf(x_proposals))

# Accept if Y <= f(X)
f_at_x = f_pdf(x_proposals)
accept_mask = y_props <= f_at_x

# Separate accepted and rejected samples for plotting
x_accept = x_proposals[accept_mask]
y_accept = y_props[accept_mask]
x_reject = x_proposals[~accept_mask]
y_reject = y_props[~accept_mask]

# Prepare curves for plotting
x = np.linspace(0, 1, 500)
f_curve = f_pdf(x)
Mg_curve = M * g_pdf(x)

# Plot
plt.figure(figsize=(7, 5))
plt.plot(x, f_curve, color='blue', lw=2, label=f'Beta({alpha},{beta_param})')
plt.plot(x, Mg_curve, color='orange', lw=2, label=f'{M} * Uniform({a},{b})')

# Draw vertical lines for uniform boundaries
plt.vlines(x=a, ymin=0, ymax=max(Mg_curve), colors='orange', linestyles='--')
plt.vlines(x=b, ymin=0, ymax=max(Mg_curve), colors='orange', linestyles='--')

# Scatter the darts under the envelope in blue (accepted) and red (rejected)
plt.scatter(x_accept, y_accept, color='blue', s=12, alpha=0.7, label='Accepted')
plt.scatter(x_reject, y_reject, color='red', s=12, alpha=0.7, label='Rejected')

# Labels and styling
plt.title('Rejection Sampling Visualization')
plt.xlabel('x')
plt.ylabel('Density')
plt.ylim(0, max(Mg_curve) * 1.1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.show()

# Print empirical acceptance rate
print(f"Acceptance rate ≈ {accept_mask.mean():.3f} (expected ~ 1/M = {1/M:.3f})")
```

Listing A.2: script that generates Figure 3.6

```
import numpy as np
```

```
import matplotlib.pyplot as plt

# Parameters
z_values = np.arange(-11, 12)  # z in {-11, ..., 11}
M = 1.11
n_samples = 2000

# Proposal g(z)
g_probs = np.zeros_like(z_values, dtype=float)
g_probs[z_values == -11] = 1/63
g_probs[z_values == 11]  = 1/63
g_probs[z_values == -10] = 2/63
g_probs[z_values == 10]  = 2/63
mask_inner = (z_values >= -9) & (z_values <= 9)
g_probs[mask_inner] = 3/63

# Target f(z)
f_probs = np.zeros_like(z_values, dtype=float)
mask_target = (z_values >= -9) & (z_values <= 9)
f_probs[mask_target] = 1/19

# ----- Discrete rejection sampling -----
x_accept, y_accept = [], []
x_reject, y_reject = [], []

for _ in range(n_samples):
    z = np.random.choice(z_values, p=g_probs)
    u = np.random.uniform(0, 1)
    y = u * M * g_probs[z_values == z][0]
    if y <= f_probs[z_values == z][0]:
        x_accept.append(z)
        y_accept.append(y)
    else:
        x_reject.append(z)
        y_reject.append(y)

# Plot
plt.figure(figsize=(10,5))

# Continuous line for proposal scaled by M
plt.plot(z_values, M*g_probs, color='orange', lw=2, marker='o',
    label=f"M*g(z)")

# Continuous line for target only within [-9, 9]
plt.plot(z_values[mask_target], f_probs[mask_target], color='blue
    ', lw=2, marker='x', label="f(z)")

# Scatter the darts in green (accepted) and red (rejected)
plt.scatter(x_accept, y_accept, color='green', s=20, alpha=0.7,
    label='Accepted')
plt.scatter(x_reject, y_reject, color='red', s=20, alpha=0.5,
    label='Rejected')

```

```
# Labels and styling
plt.title("Rejection Sampling with discrete Uniform target distribution")
plt.xlabel("z")
plt.ylabel("Density")
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.show()

# Print empirical acceptance rate
acceptance_rate = len(x_accept)/n_samples
print(f"Acceptance rate ≈ {acceptance_rate:.3f} (expected ~ 1/M = {1/M:.3f})")
```

Listing A.3: script that generates Figure 3.8

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import uniform, truncnorm, triang

# Support
x = np.linspace(-8, 8, 2000)

# --- Proposal distributions and their samplers ---

rng = np.random.default_rng(123) # RNG for sampling output values

# 1. Uniform(-8, 8)
g1 = uniform.pdf(x, loc=-8, scale=16)
sampler1 = lambda: rng.uniform(-8, 8)

# 2. Truncated Gaussian N(0, 4.5^2) in [-8, 8]
sigma = 4.5
a, b = -8/sigma, 8/sigma
g2 = truncnorm.pdf(x, a, b, loc=0, scale=sigma)
sampler2 = lambda: truncnorm.rvs(a, b, loc=0, scale=sigma, random_state=rng)

# 3. Triangular(-8, 8, mode=0)
# scipy's triang takes c = (mode - loc)/scale
c = (0 - (-8)) / 16
g3 = triang.pdf(x, c=c, loc=-8, scale=16)
sampler3 = lambda: rng.triangular(-8, 0, 8)

# --- Draw samples directly from proposals (no rejection sampling) ---
samples1 = [sampler1() for _ in range(2000)]
samples2 = [sampler2() for _ in range(2000)]
samples3 = [sampler3() for _ in range(2000)]

# --- Plot results ---
plt.figure(figsize=(14, 4))

# Proposal 1
```

```
plt.subplot(1, 3, 1)
plt.hist(samples1, bins=30, density=True, alpha=0.6, color="orange", label="Samples from g1")
plt.plot(x, g1, color="black", lw=2, label="PDF g1")
plt.title("Uniform distribution")
plt.legend()

# Proposal 2
plt.subplot(1, 3, 2)
plt.hist(samples2, bins=30, density=True, alpha=0.6, color="green", label="Samples from g2")
plt.plot(x, g2, color="black", lw=2, label="PDF g2")
plt.title("Truncated Gaussian distribution")
plt.legend()

# Proposal 3
plt.subplot(1, 3, 3)
plt.hist(samples3, bins=30, density=True, alpha=0.6, color="red", label="Samples from g3")
plt.plot(x, g3, color="black", lw=2, label="PDF g3")
plt.title("Triangular distribution")
plt.legend()

plt.tight_layout()
plt.show()
```

Listing A.4: script that generates Figure 3.9

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import uniform, norm, triang

# Support
x = np.linspace(-8, 8, 2000)

# --- Proposal distributions ---

# 1. Uniform(-8, 8)
g1 = uniform.pdf(x, loc=-8, scale=16)

# 2. Truncated Gaussian N(0, 4.5^2) in [-8, 8]
sigma = 4.5
g2 = norm.pdf(x, loc=0, scale=sigma)
# normalize over [-8,8]
Z2 = np.trapezoid(g2, x)
g2 /= Z2

# 3. Triangular(-8, 8, mode=0)
# scipy's triang takes c = (mode - loc)/scale
c = (0 - (-8)) / 16
g3 = triang.pdf(x, c=c, loc=-8, scale=16)

# --- Target distribution: Raised Cosine ---
f = (5 + 5 * np.cos(np.pi * x / 8))
```

```
27
28 f /= np.trapezoid(f, x)  # normalize to integrate to 1
29
30 # --- Envelopes (constants chosen manually after empirical
       evaluation)---
31 M1, M2, M3 = 2.05, 1.35, 1.2
32 Mg1, Mg2, Mg3 = M1 * g1, M2 * g2, M3 * g3
33
34 # --- Plot ---
35 plt.figure(figsize=(8,5))
36
37 plt.plot(x, Mg1, label="Proposal g1: Uniform(-8,8)", color="
       orange")
38 plt.plot(x, Mg2, label="Proposal g2: Truncated Gaussian(0, 4.5)",
        color="green")
39 plt.plot(x, Mg3, label="Proposal g3: Triangular(-8,8, mode=0)",
       color="red")
40 plt.plot(x, f, label="Target f: Raised Cosine(μ=0, s=8)", color="
       blue", lw=2)
41
42 plt.title("Scaled Proposal Distributions and Target Distribution"
       )
43 plt.xlabel("x")
44 plt.ylabel("Density")
45 plt.legend()
46 plt.grid(alpha=0.3)
47 plt.show()
```

Listing A.5: script that generates Figure 3.10

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from scipy.stats import triang, truncnorm
4
5  # Support
6  x = np.linspace(-8, 8, 2000)
7
8  # --- Target distribution: Raised Cosine ---
9  f = 5 + 5 * np.cos(np.pi * x / 8)
10 f /= np.trapezoid(f, x)  # normalize to integrate to 1
11
12 # --- Envelopes (constants chosen manually after empirical
       evaluation) ---
13 M1, M2, M3 = 2.05, 1.35, 1.2
14
15 # --- Rejection sampling procedure ---
16 rng = np.random.default_rng(123) # RNG for sampling output values
17
18 def rejection_sampling(pdf, sampler, M, f, x_support, n_samples
       =2000):
19     accepted = []
20     while len(accepted) < n_samples:
21         # sample from proposal
22         z = sampler()
```

```
        # evaluate densities
        g_z = pdf(z)
        f_z = np.interp(z, x_support, f)
        # sample from uniform(0, M*g(z))
        u = rng.uniform(0, M * g_z)
        if u <= f_z:
            accepted.append(z)
    return np.array(accepted)

# --- Sampler and PDF for each proposal distribution ---

# 1. Uniform(-8, 8)
sampler1 = lambda: rng.uniform(-8, 8)
pdf1 = lambda z: 1/16 if -8 <= z <= 8 else 0

# 2. Truncated Gaussian N(0, 4.5^2) in [-8, 8]
sigma = 4.5
a, b = -8/sigma, 8/sigma  # standardized bounds
sampler2 = lambda: truncnorm.rvs(a, b, loc=0, scale=sigma,
   random_state=rng)
pdf2 = lambda z: truncnorm.pdf(z, a, b, loc=0, scale=sigma)


#3. Triangular(-8, 8, mode=0)
# scipy's triang takes c = (mode - loc)/scale
c = (0 - (-8)) / 16  # mode at 0
sampler3 = lambda: rng.triangular(left=-8, mode=0, right=8)
pdf3 = lambda z: triang.pdf(z, c=c, loc=-8, scale=16)

# --- Perform rejection sampling step ---
samples1 = rejection_sampling(pdf1, sampler1, M1, f, x, n_samples
   =2000)
samples2 = rejection_sampling(pdf2, sampler2, M2, f, x, n_samples
   =2000)
samples3 = rejection_sampling(pdf3, sampler3, M3, f, x, n_samples
   =2000)

# --- Plot results ---
plt.figure(figsize=(14, 4))

# Proposal 1
plt.subplot(1, 3, 1)
plt.hist(samples1, bins=30, density=True, alpha=0.6, color="
   orange", label="Samples from g1")
plt.plot(x, f, color="blue", lw=2, label="Target f(x)")
plt.title("Rejection Sampling with Proposal g1 (Uniform)")
plt.legend()

# Proposal 2
plt.subplot(1, 3, 2)
plt.hist(samples2, bins=30, density=True, alpha=0.6, color="green
   ", label="Samples from g2")
plt.plot(x, f, color="blue", lw=2, label="Target f(x)")
```

```
plt.title("Rejection Sampling with Proposal g2 (Truncated
    Gaussian)")
plt.legend()

# Proposal 3
plt.subplot(1, 3, 3)
plt.hist(samples3, bins=30, density=True, alpha=0.6, color="red",
     label="Samples from g3")
plt.plot(x, f, color="blue", lw=2, label="Target f(x)")
plt.title("Rejection Sampling with Proposal g3 (Triangular)")
plt.legend()

plt.tight_layout()
plt.show()
```

Listing A.6: script that generates Figure 3.11

A.2 Scripts used for Chapter 4

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import math

# Parameters
v = 10
alpha = 12 #=2 for Figure 4.2 or =4 for Figure 4.4 or =12 for
    Figure 4.5
sigma = alpha*v
M = math.e  # scaling factor

# X range wide enough to capture both distributions
x = np.linspace(-5*sigma, 10*sigma, 5000)

# Compute PDFs
pdf_centered = norm.pdf(x, loc=0, scale=sigma)
pdf_shifted_scaled = M * norm.pdf(x, loc=-v, scale=sigma)

# Plot Gaussian centered at 0
plt.plot(x, pdf_centered, color="red", lw=2,
         label=rf"Target $D_{{0,{sigma}}}$")

# Plot scaled Gaussian centered at -v
plt.plot(x, pdf_shifted_scaled, color="black", lw=2,
         label=rf"$Scaled \ M \cdot D_{{-{v},{sigma}}}$")

# Vertical guide lines at 0 and -v
for c in [0, -v]:
    plt.axvline(c, color="black", lw=1, ls=":")

# Axis styling
plt.xlabel("x", fontsize=12)
```

```
plt.ylabel("Density", fontsize=12)
plt.title(rf"Gaussian distributions with $v={v},\ \sigma={sigma},\ M=e$", fontsize=13)

# Show x values and density values
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)

# Custom label for -v to not overlap with 0
ymin, ymax = plt.ylim()

#plt.text(-v-10, ymin - 0.05*(ymax-ymin), r"$-v$",
#         ha="center", va="bottom", fontsize=11) #uncomment if using alpha = 2 or 4

plt.text(-v-40, ymin - 0.05*(ymax-ymin), r"$-v$",
         ha="center", va="bottom", fontsize=11) #uncomment if using alpha = 12

plt.legend(fontsize=11, frameon=True)
plt.grid(False)
plt.tight_layout()
plt.show()
```

Listing A.7: script that generates Figure 4.2a, Figure 4.4a and Figure 4.5

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import math

# Parameters
v = 10
alpha = 1/math.sqrt(2)
sigma = alpha * v
M = math.e  # scaling factor

# X range wide enough to capture both distributions
x = np.linspace(-5 * sigma, 5 * sigma, 5000)

# Compute PDFs
pdf_target = norm.pdf(x, loc=0, scale=sigma)
pdf_proposal = 0.5 * norm.pdf(x, loc=-v, scale=sigma) + \
               0.5 * norm.pdf(x, loc=+v, scale=sigma)
pdf_shifted_scaled = M * pdf_proposal

# Plot Gaussian centered at 0
plt.plot(x, pdf_target, color="red", lw=2,
         label=rf"Target $D_{{0,\sigma}}$")

# Plot scaled bimodal with modes in -v and +v
plt.plot(x, pdf_shifted_scaled, color="black", lw=2,
         label=rf"$Scaled \ M \cdot (\frac{{1}}{{2}} D_{{-v, \sigma}} + \frac{{1}}{{2}} D_{{v,\sigma}})$")
```

```
28
29 # Vertical guide lines at 0, -v, +v
30 for c in [0, -v, v]:
31     plt.axvline(c, color="black", lw=1, ls=":")
32
33 # Axis styling
34 plt.xlabel("x", fontsize=12)
35 plt.ylabel("Density", fontsize=12)
36 plt.title(rf"The scaled bimodal Gaussian distribution ($v={v}$, $
       \sigma={sigma:.3f},\ M=e$)", fontsize=13)
37
38 # Show x values and density values
39 plt.xticks(fontsize=10)
40 plt.yticks(fontsize=10)
41
42 # Custom labels for -v and +v
43 ymin, ymax = plt.ylim()
44 plt.text(-v, ymin + 0.001, r"$-v$",
45          ha="center", va="bottom", fontsize=11)
46 plt.text(+v, ymin + 0.001, r"$+v$",
47          ha="center", va="bottom", fontsize=11)
48
49 # Legend
50 plt.legend(fontsize=11, frameon=True)
51 plt.grid(False)
52 plt.tight_layout()
53 plt.show()
```

Listing A.8: script that generates Figure 4.7

Appendix B

Future thesis work

Our goal in this appendix is to discuss the identification scheme developed by Lyubashevsky, because even though it can be transformed into a digital signature scheme, the scheme does not really satisfy all the properties of a Σ-protocol. The identification scheme has been recently discussed by Lyubashevsky, in the context of zero-knowledge proofs, at the Simons Institute for Theory of Computing in Berkeley, California (2022) [14] and in a workshop at the Institute Henri Poincaré in Paris, France (2024) [15].

B.1 Lyubashevsky's lattice-based identification protocol

The sketch of the identification scheme is shown in Figure B.1 we consider all the elements of the scheme over the ring $\mathbb{Z}_q[X]/(X^n+1)$. The secret key is an element s such that $As = t \bmod q$ and $||s||$ is small, where the pair (A, t) is the public key.

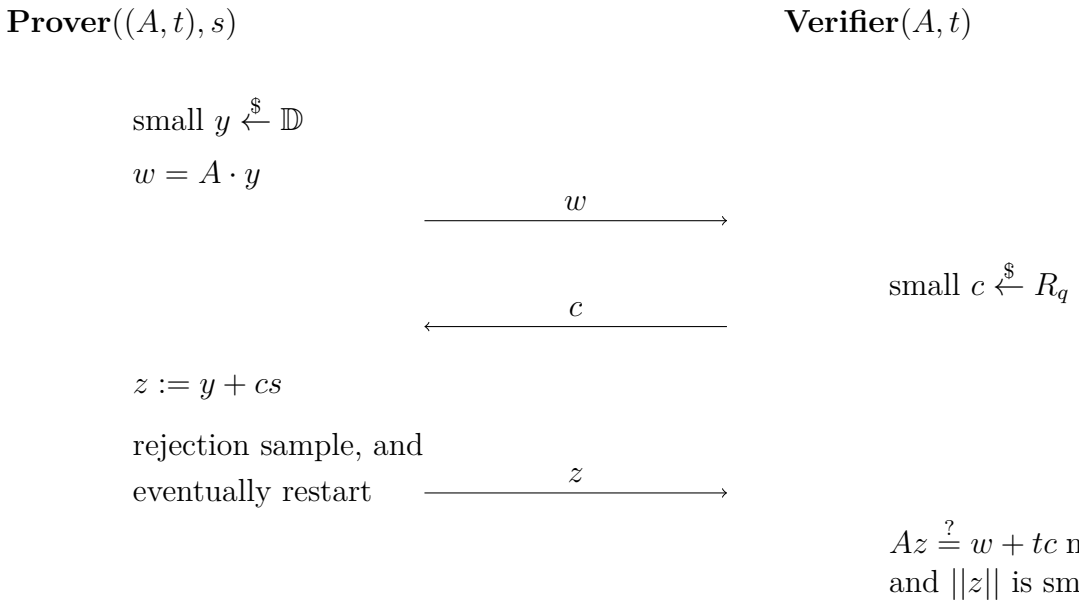


Figure B.1: Lyubashevsky's lattice-based identification protocol

B.1.1 The limit of extractability in Lyubashevsky's identification scheme

We can see that this identification protocol is quite similar to the Schnorr identification protocol, in fact Lyubashevsky was inspired by Schnorr's identification scheme to build his lattice-based one. Despite the similarity, Lyubashevsky's identification scheme doesn't fully satisfy the properties of a Σ-protocol. Let's check what is the matter with the properties of a Σ-protocol.

Completeness. If the prover and verifier behave honestly, the verifier always accepts. Indeed, $Az = Ay + Asc = w + tc$ since $w = Ay$ and $t = As$, and the norm of $z = sc + y$ is small because s, c are small and y is a small polynomial drawn according to the distribution $\mathbb{D}$.

Honest-Verifier Zero-Knowledge. The interaction between the prover and the verifier does not leak any information about the secret: as we know, thanks to rejection sampling, the response z is made independent of the secret s

Special Soundness. This is the property that it's not fully achieved. Let's see what can we extract given two valid transcripts (w, c, z) and (w, c', z') with $c \neq c'$. Since the verifier accepts the transcripts we can write $Az = w + tc \bmod q$ and $Az' = w + tc' \bmod q$, then we can subtract the first identity from the second identity and obtain $A(z'-z) = t(c-c') \bmod q$. We can rewrite it as $A\overline{z} = t\overline{c} \bmod q$.

And that's where we stop since we can't really divide $\overline{z}$ by $\overline{c}$, because the result it's not guaranteed to be small. Thus, we can only say that we prove the knowledge of a small $\overline{z}$ such that $A\overline{z} = t\overline{c} \bmod q$ for some $\overline{c}$, but $||z|| \geq ||s||$.

Such proofs of knowledge are good enough for constructing basic protocols like digital signatures, but they are not sufficient for building more complex protocols for which you are required to prove the knowledge of some function of the secret itself. Let's explain this more in details.

Consider the case of the basic digital signature protocol. Proving knowledge of $(\overline{z}, \overline{c})$ is enough because the security relies on the unforgeability of a valid signature: we care that an adversary cannot forge a valid signature without knowledge of a valid witness. So if an adversary is able to forge a valid signature (z, c) and he can produce another signature (z', c') for the same message then, as we know from the security reduction in figure 3.13, he can set $\overline{z} = z - z'$ and $\overline{c} = c - c'$ and obtain a SIS solution as $A\overline{z} + t\overline{c} = 0$. Thus, by proving knowledge of a valid witness $(\overline{z}, \overline{c})$ such that $A\overline{z} = t\overline{c}$ you prove that you can really create valid signatures, otherwise it would mean that you can solve SIS, which is impossible.

But for other applications, like verifiable encryption, it's not enough [16]. Suppose that $As = t$ is an encryption where t is the ciphertext and s is the message (with randomness), and we can only prove knowledge of some $(\overline{z}, \overline{c})$ such that $A\overline{z} = t\overline{c}$. In order to decrypt, it is necessary for t to have a short preimage, so proving knowledge of $(\overline{z}, \overline{c})$ is not enough to guarantee that the ciphertext t can be decrypted because it is $t\overline{c}$ that has a short preimage, not t, and $\overline{c}$ is not known by the decryptor. A consequence of this is that the currenlty most efficient verifiable encryption scheme [17] has an expected decryption time

equal to the one of an adversary who tries to break the scheme because the decryptor essentially need to guess c to recover $s = \overline{z}/\overline{c}$.

Thus proving knowledge of $(\overline{z}, \overline{c})$ it's not good enough for actual proofs of knowledge because eventually you want to prove that you have some function $f(s)$ of s, meaning that you really need to prove knowledge of s as the short preimage of $f(s)$.

B.1.2 The need for a commitment scheme

To solve that issue we need to rely on a commitment scheme that allow us to prove properties about the committed values, in particular the short secret s.

Indeed, Lyubashevsky develops a new general commitment scheme called ABDLOP commitment scheme, which is used along with techniques such as inner products, linear proofs over $\mathbb{Z}_q$, quadratic proofs over $\mathbb{Z}_q$, approximate range proofs and the Johnson-Lindenstrauss lemma, to provide an efficient framework for lattice-based zero-knowledge proofs.

Then, with the intention of continuing to study Lyubashevsky's techniques and frameworks, one could go into detail of the work "Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General" [16].

Appendix C

Video timestamps

C.1 Talks about digital signatures [5] [6]

C.1.1 Timestamps of the lecture at the UCI Workshop (2013)

Link: `https://www.youtube.com/watch?v=D-IQcoj0kdo`

- 07:40 - the Ring-SIS problem
- 13:00 - the identification scheme
- 20:40 - the secret and the response must be independent
- 24:00 - proof of security of the identification scheme
- 31:30 - the idea of rejection sampling
- 38:00 - rejection sampling over the coefficients
- 46:30 - rejection sampling with Gaussian distribution
- 54:00 - the bimodal Gaussian idea

C.1.2 Timestamps of the lecture at Microsoft (2016)

Link: `https://www.youtube.com/watch?v=F9RQ-SxtkBQ`

- 01:00 - lattice hard problems
- 11:00 - the GVP hash and sign signature
- 23:30 - fiat-shamir digital signature and proof of security
- 34:00 - why we want to based our digital signature on both SIS and LWE
- 37:00 - rejection sampling with Gaussian distribution
- 47:00 - rejection sampling with bimodal Gaussian idea

C.2 Talks about zero-knowledge proofs [14] [15]

C.2.1 Timestamps of the lecture at Simons Institute (2022)

Link: `https://www.youtube.com/watch?v=xEDZ4tyesMY`

- 08:00 - recall Schnorr identification scheme
- 15:30 - why we don't prove exactly to know the short vector s, analysis of soudness property
- 19:00 - the proof is enough for digital signature schemes, but not for non-basic schemes like encryption
- 22:55 - the necessity of a commitment scheme
- 33:40 - recall the Ajtai and the BDLOP commitment schemes
- 40:00 - the (new) ABDLOP commitment scheme
- 44:00 - inner products
- 47:00 - proving knowledge of the constant coefficient of a linear function over R_q (is equivalent to proving linear relations over $\mathbb{Z}_q$)
- 54:00 - proving linear functions over $\mathbb{Z}$
- 54:00 - proving quadratic relations

C.2.2 Timestamps of the workshop at Institute Henri Poincaré (2024)

Link: `https://www.carmin.tv/en/video/lattices-and-zero-knowledge`

- 08:30 - presentation of the lattice based identification scheme as a zero knowledge proof of the secret key s such that As = t
- 14:30 - we don't prove exactly knowledge of s such that As = t
- 15:00 - the Ajtai commitment scheme
- 16:30 proving linear relations over $\mathbb{Z}_p[X]/(X^n + 1)$
- 35:00 - proving quadratic relations over $\mathbb{Z}_p[X]/(X^n + 1)$
- 41:30 - proving knowledge of s such that As=t and $||s|| = \beta$, for some value β.
- 46:00 - succint proofs

Bibliography

[1] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. Version 0.6. Last accessed: 2025-09-27. Jan. 2023. URL: https://toc.cryptobook.us/book.pdf (cit. on p. 12).

[2] Vadim Lyubashevsky. *Lattice Signatures Without Trapdoors*. Cryptology ePrint Archive, Paper 2011/537. 2011. URL: https://eprint.iacr.org/2011/537 (cit. on pp. 13, 14, 24–27, 29, 30, 34).

[3] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. *Trapdoors for Hard Lattices and New Cryptographic Constructions*. Cryptology ePrint Archive, Paper 2007/432. 2007. URL: https://eprint.iacr.org/2007/432 (cit. on p. 13).

[4] Vadim Lyubashevsky. «Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures». In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 598–616. DOI: 10.1007/978-3-642-10366-7_35. URL: https://www.iacr.org/archive/asiacrypt2009/59120596/59120596.pdf (cit. on p. 13).

[5] Workshop on Lattices with Symmetry. *15 Vadim Lyubashevsky on Lattice-Based Digital Signatures*. YouTube, Accessed: 2025-08-22. 2013. URL: https://www.youtube.com/watch?v=D-IQcoj0kdo (cit. on pp. 13, 39, 71).

[6] Microsoft Research. *Lattice Signatures Schemes*. YouTube, Accessed: 2025-08-22. 2016. URL: https://www.youtube.com/watch?v=F9RQ-SxtkBQ (cit. on pp. 13, 35, 39, 71).

[7] Roger Eckhardt. «Stan Ulam, John von Neumann, and the Monte Carlo Method». In: *Los Alamos Science* 15 (1987). Accessed: 2025-10-10, pp. 131–137. URL: https://mcnp.lanl.gov/pdf_files/Article_1987_LAS_Eckhardt_131--141.pdf (cit. on p. 15).

[8] Kapil Sachdeva. *Rejection Sampling - VISUALLY EXPLAINED with EXAMPLES!* YouTube, Accessed: 2025-08-22. 2021. URL: https://www.youtube.com/watch?v=si76S7QqxTU (cit. on p. 15).

[9] Kapil Sachdeva. *What is Rejection Sampling?* Accessed: 2025-08-22. 2021. URL: https://medium.com/data-science/what-is-rejection-sampling-1f6aff92330d (cit. on p. 15).

[10] Gregory Gundersen. *Sampling: Two Basic Algorithms*. Accessed: 2025-08-22. 2019. URL: https://gregorygundersen.com/blog/2019/09/01/sampling/ (cit. on p. 19).

[11] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. *Lattice Signatures and Bimodal Gaussians*. Cryptology ePrint Archive, Paper 2013/383. 2013. URL: https://eprint.iacr.org/2013/383 (cit. on pp. 20, 39, 44, 45, 47–51, 53).

[12] Vadim Lyubashevsky. «Lattice-Based Identification Schemes Secure Under Active Attacks». In: *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*. Vol. 4939. Lecture Notes in Computer Science. Springer, 2008, pp. 162–179. DOI: 10.1007/978-3-540-78440-1_10. URL: https://iacr.org/archive/pkc2008/49390163/49390163.pdf (cit. on p. 20).

[13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. *BLISS Signature Scheme — Implementation*. Accessed: 2025-09-28. 2013. URL: https://bliss.di.ens.fr/ (cit. on p. 51).

[14] Simons Institute. *Lattice-based Zero-knowledge Proofs*. YouTube, Accessed: 2025-08-22. 2022. URL: https://www.youtube.com/watch?v=xEDZ4tyesMY (cit. on pp. 67, 72).

[15] Vadim Lyubashevsky. *Lattices and Zero-Knowledge*. Audiovisual resource, Institut Henri Poincaré (IHP). Nov. 2024. DOI: 10.57987/IHP.2024.T3.WS2.005. URL: https://dx.doi.org/10.57987/IHP.2024.T3.WS2.005 (cit. on pp. 67, 72).

[16] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. *Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General*. Cryptology ePrint Archive, Paper 2022/284. 2022. URL: https://eprint.iacr.org/2022/284 (cit. on pp. 68, 69).

[17] Vadim Lyubashevsky and Gregory Neven. *One-Shot Verifiable Encryption from Lattices*. Cryptology ePrint Archive, Paper 2017/122. 2017. URL: https://eprint.iacr.org/2017/122 (cit. on p. 68).