

# Post-Quantum Cryptography: Lattice Signatures

Johannes Buchmann      Richard Lindner  
Markus Rückert      Michael Schneider

Technische Universität Darmstadt  
Department of Computer Science  
Hochschulstraße 10, 64289 Darmstadt, Germany

## Abstract

This survey provides a comparative overview of lattice-based signature schemes with respect to security and performance. Furthermore, we explicitly show how to construct a competitive and provably secure Merkle-tree signature scheme solely based on worst-case lattice problems.

**Keywords:** digital signatures, post-quantum cryptography, lattices.

## 1 Introduction

In the advent of quantum computers, today’s widespread signature schemes — most importantly RSA and DSA — are rendered utterly insecure due to the seminal work of Shor [33]. Digital signatures, however, have become a “supporting pillar” of the world’s economy. Thus, endangering their security results in a potential collapse of electronic commerce and secure internet communication as a whole.

Due the conjectured intractability of lattice problems, like approximating the shortest lattice vector (SVP), even in the quantum-era and because of their computational efficiency, lattice-based signature schemes seem to be one of the most promising replacements for current constructions.

With our work, we compare existing alternatives with respect to security and efficiency, which leads us to proposing a Merkle-tree construction in

Scheme	Security proof	Signature size	Verification-key size	Signing-key size	Complexity assumption
GGH	broken	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n^2 \log(n))$	$\mathcal{O}(n^2 \log(n))$	(CVP)
NTRUSign	broken	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n \log(n))$	(CVP)
GMSS-SWIFFT-W	yes	$\mathcal{O}(n^2 \log^2(n))$	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n^2 \log^2(n))$	worst-case ISVP <sub>n</sub>
GMSS-SWIFFT-LM	yes	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n^2 \log^2(n))$	worst-case ISVP <sub>n</sub>
GPV	yes (RO)	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n^2 \log(n))$	$\mathcal{O}(n^2 \log(n))$	worst-case SIVP <sub>n</sub>

SIVP<sub>n</sub> is the shortest independent vector problem in dimension  $n$ , ISVP<sub>n</sub> is the ideal shortest vector problem in dimension  $n$ . GMSS-SWIFFT-LM denotes GMSS with Winternitz’ OTS and SWIFFT as the hash function. GMSS-SWIFFT-LM denotes GMSS with SWIFFT in conjunction with LM as the OTS. “RO” denotes the random oracle model. For detailed figures, we refer the reader to the respective section below.

Table 1: Comparison of signature and key sizes for lattice-based schemes.

conjunction with a lattice-based one-time signature scheme (OTS) and a lattice-based hash function as the preferred choice. In our comparison, we consider the signature scheme of Gentry, Peikert, and Vaikuntanathan (GPV) [12], Lyubashevsky and Micciancio’s OTS (LM) [21], as well as Winternitz’ OTS (W) [23] with Lyubashevsky et al.’s hash function (SWIFFT) [22].

Note that all of the above schemes are provably secure and their security is based on worst-case lattice problems due to a famous result by Ajtai [1], which puts lattice-based cryptography in an almost unique position of requiring exceptionally weak assumptions. Whereas the complexity assumption for Winternitz with SWIFFT and LM (SVP in ideal lattices) is probably stronger than that for GPV (SVP in general lattices), the latter is proven in the weaker random oracle model. Earlier approaches, like GGH [13] and NTRUSign [14], were fully broken despite the common belief (*not proof*) that one had to solve the closest vector problem in a lattice for this to happen (cf. Section 2.3).

Applying GMSS [5], an improvement of Merkle’s generic construction principle [23], turns both one-time signature schemes into *multi-time* signature schemes, allowing a comparison with GPV in Table 1. Observe that the GMSS schemes have smaller verification keys than GPV. Considering the signature size, we expect GMSS-SWIFFT-LM to be similar to GPV and better than GMSS-SWIFFT-W.

Since asymptotics can be deceptive, we take a closer look at the three

Scheme	Worst-case problem dimension $n$	Signature size (KB)	Verification-key size (KB)	Signing-key size (KB)
GMSS-SWIFFT-W	64	12.52	0.06	19.29
	128	44.34	0.13	51.48
GMSS-SWIFFT-LM	64	2.79	0.06	12.79
	128	5.67	0.13	25.63
GPV	64	13.18	6.12	6.12
	128	33.75	33.30	33.30
RSA-1024	-	0.13	0.16	0.62
RSA-2048	-	0.25	0.29	1.19

We estimate the sizes of signatures and keys under the assumption that solving the worst-case lattice problem in dimension  $n$  is hard. The first dimension  $n = 64$  is chosen according to the proposed parameters for SWIFFT, which are conjectured to provide 100 bit-security [22]. The second dimension  $n = 128$  should provide an outlook on future developments and how they might affect efficiency. For the GMSS sizes we used the parameter set  $\{2, (20, 20), (10, 5)\}$ , the output length of the hash function SWIFFT is 513 bit for  $n = 64$  and 1026 bit for  $n = 128$ , respectively. See Section 3 for details.

Table 2: Explicit signature and key sizes.

candidates in Table 2, where we estimate explicit signature and key sizes based on recommendations in [22]. From this table, we infer that GMSS-SWIFFT-LM is currently the most efficient lattice-based signature scheme. Its verification key size is even superior to RSA. Although its signature key is larger than that of GPV at normal security level, this situation is reversed for higher security levels. Thus, we consider GMSS-SWIFFT-LM secure and practical even for larger security parameters.

**Organization.** After recalling some basic definitions and notations, we chronologically discuss the various lattice based signature schemes, starting with GGH and NTRUSign in Section 2. Section 3 is devoted to hash-based signature schemes according to Merkle and their provably secure realization based on worst-case lattice problems. Finally, Section 4 discusses the GPV signature scheme and its security.

## 1.1 Preliminaries

Throughout the paper,  $n$  denotes the security parameter.

**Digital signatures.** A digital signature scheme DS is a triple  $(\text{Kg}, \text{Sig}, \text{Vf})$  where  $\text{Kg}(n)$  outputs a private signing key  $sk$  and a public verification key  $pk$ ;  $\text{Sig}(sk, M)$  outputs a signature  $\sigma$  on a message  $M$  from the message space  $\mathcal{M}$  under  $sk$ ;  $\text{Vf}(pk, \sigma, M)$  outputs 1 if  $\sigma$  is a valid signature on  $M$  under  $pk$  and otherwise 0.

Signature schemes are complete if for all  $(sk, pk) \leftarrow \text{Kg}(n)$ , all messages  $M \in \mathcal{M}$ , and any  $\sigma \leftarrow \text{Sig}(sk, M)$ , we have  $\text{Vf}(pk, \sigma, M) = 1$ .

Security of digital signature schemes is typically proven against existential forgery under a chosen message attack (EU-CMA), where an adversary wins if he outputs a signature on a *new* message  $M^*$  after accessing a signature oracle on a polynomial number of different messages. For the described constructions, we need the notion of strong unforgeability under a chosen message attack (SU-CMA), where the adversary even wins if he is able to output a *new pair*  $(M^*, \sigma^*)$ , i.e. he is not forced to output a signature on a new message. In the random oracle model, the adversary has access to a hash oracle  $H$  that is chosen from the family of all collision resistant hash functions  $\mathcal{H}(n)$ . The described concept is formalized in the following experiment.

**Experiment**  $\text{Exp}_{\mathcal{A}, \text{DS}}^{\text{su-cma}}(n)$

$H \leftarrow \mathcal{H}(n)$

$(sk, pk) \leftarrow \text{Kg}(n)$

$(M^*, \sigma^*) \leftarrow \mathcal{A}^{H(\cdot), \text{Sig}(sk, \cdot)}(pk)$

let  $\sigma_i$  be the answer returned by  $\text{Sig}(sk, \cdot)$  on input  $M_i$ , for  $i = 1, \dots, k$ .

Return 1 iff  $\text{Vf}(pk, M^*, \sigma^*) = 1$  and  $(M^*, \sigma^*) \notin \{(M_1, \sigma_1), \dots, (M_k, \sigma_k)\}$ .

The scheme DS is called  $(t, q_{\text{Sig}}, q_H, \epsilon)$ -strongly unforgeable if there is no adversary, running in time at most  $t$  while making at most  $q_{\text{Sig}}$  queries to the oracle  $\text{Sig}(sk, \cdot)$  and at most  $q_H$  queries to the hash oracle, that succeeds in the above experiment with probability at least  $\epsilon$ .

**Lattices.** A lattice is a set  $\Lambda = \{\sum_{i=1}^d x_i \mathbf{b}_i \mid x_i \in \mathbb{Z}\}$  in  $\mathbb{R}^n$ ,  $\mathbf{b}_1, \dots, \mathbf{b}_d$  are linearly independent over  $\mathbb{R}$ . The matrix  $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_d]$  is a *basis* of the lattice  $\Lambda$  and we write  $\Lambda = \Lambda(\mathbf{B})$ . The number of vectors in the basis is the dimension of the lattice. For each lattice there is a unique basis  $\mathbf{B}$  for which the following conditions hold: 1.  $b_{ij} = 0$  for  $i > j$ , i.e.  $\mathbf{B}$  is upper triangular; 2.  $b_{ii} > 0$  for  $i = 1, \dots, d$ ; 3.  $0 \leq b_{ij} < b_{ii}$  for  $i < j$ . Any such basis is in Hermite normal form. This form can be computed efficiently for any basis and is denoted  $\text{HNF}(\mathbf{B})$ .

Now, consider *modular lattices* as a special form of lattices. Given a modulus  $q$ , a matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ , and the equation

$$\mathbf{A} \mathbf{v} \equiv \mathbf{0} \pmod{q}, \quad (1)$$

then the set of all vectors  $\mathbf{v} \in \mathbb{Z}_q^m$  that satisfy the above equation is a lattice. Lattices of this form are denoted with  $\Lambda_q^\perp(\mathbf{A})$ .

Consider the ring  $R = \mathbb{Z}[x]/\langle v(x) \rangle$ , where  $v$  is a monic polynomial of degree  $n$ . A lattice  $\Lambda$  is an *ideal lattice*, if there exists an ideal  $I \subseteq R$ , such that  $I = \{a_0 + \dots + a_{n-1}x^{n-1} : \mathbf{a} \in \Lambda\}$ . The ideal lattices of the ring  $\mathbb{Z}[x]/\langle x^n - 1 \rangle$  are *cyclic lattices*.

The main computational problem in lattices is the (approximate) shortest vector problem (SVP), where an algorithm is given a description, a basis, of a lattice  $\Lambda$  and is supposed to find the shortest vector  $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$  with respect to a certain  $\ell_p$  norm (up to an approximation factor). More precisely, find a vector  $\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}$ , such that  $\|\mathbf{v}\|_p \leq \gamma \|\mathbf{w}\|_p$  for all  $\mathbf{w} \in \Lambda \setminus \{\mathbf{0}\}$  for a fixed approximation factor  $\gamma \geq 1$ . This problem with a constant approximation factor is known to be  $\mathcal{NP}$ -hard for all  $\ell_p$  norms [9, 32, 17]. For approximation factors which are exponential in the lattice dimension, the problem is solvable in polynomial time with the famous LLL algorithm by Lenstra, Lenstra, and Lovász [19]. We refer the reader interested in the currently known “approximability” and “inapproximability” results to a recent survey [31] by Regev.

The same problem in *ideal lattices* is the (approximate) ideal shortest vector problem (ISVP) [20]. This problem is not considered to be  $\mathcal{NP}$ -hard. However, no known lattice basis reduction method is significantly more efficient on ideal lattices than on general ones. The latest work on reducing ideal lattice bases was by Gama and Nguyen [10].

Another lattice problem is the (approximate) shortest independent vector problem (SIVP) [26], where we are given a basis of the lattice  $\Lambda$  and asked to find  $n$  linearly independent vectors  $\mathbf{s}_1, \dots, \mathbf{s}_n \in \Lambda$ , such that  $\|\mathbf{s}_i\| \leq \gamma \lambda_n(\Lambda)$  for all  $i = 1, \dots, n$ . Here  $\lambda_n(\Lambda)$  is the radius of the smallest sphere centered in the origin containing  $n$  linearly independent lattice vectors and  $\gamma \geq 1$  the usual approximation factor.

In the case of modular lattices, there is also a special version of the SVP, named short integer solution problem. We denote it with  $\text{SIS}(q, m, \beta)$ . There, an algorithm is given a basis of  $\Lambda_q^\perp(\mathbf{A})$  and is supposed to output a non-zero solution  $\mathbf{v} \in \mathbb{Z}_q^m$  to (1). The algorithm succeeds if  $\|\mathbf{v}\|_2 \leq \beta$  for a given

norm bound  $\beta$ . Replacing the right-hand side  $\mathbf{0}$  in (1) with  $\mathbf{y}$ , yields the inhomogeneous case of SIS, denoted with  $\text{ISIS}(q, m, \beta)$ . The SIS (ISIS) is  $(t, \epsilon)$ -hard if there is no algorithm running in time at most  $t$  that solves the SIS (ISIS) with probability at least  $\epsilon$ . The problems SIS and ISIS were, in principle, introduced by Ajtai [1] and its hardness is analyzed in [27] and [12]. The latter work also explicitly deals with the  $\ell_\infty$  norm, which we will use in our security proofs.

## 2 Signatures based on CVP

### 2.1 GGH

The first proposal for a lattice-based signature was given at CRYPTO '97 by Goldreich, Goldwasser, and Halevi [13]. Their idea was to use a lattice for which a “bad” basis, whose vectors are long and almost parallel is public, and a “good” basis with short and nearly orthogonal vectors is private. To employ their scheme, messages need to be hashed into the space spanned by the lattice, and the signature for a given hash in this space is the *closest* lattice point. The scheme did not come with a formal proof of security and was broken in its basic variant in 1999 by Nguyen [28]. In 2006 a modified variant was broken again by Nguyen and Regev [29] (cf. Section 2.3).

We will give a formal description of the scheme. The security parameter is  $n$ . We will describe the key generation with security improvements due to Micciancio [24].

**Key generation.** The main security parameter is an integer  $n$ . Both the public key  $\mathbf{B}$  and private key  $\mathbf{R}$  in the scheme are matrices in  $\mathbb{Z}^{n \times n}$ . Two distributions for choosing the private key were suggested by Goldreich et al. [13]

- “random lattice”: Choose  $\mathbf{R}$  uniformly at random from  $\{-l, \dots, l\}^{n \times n}$ , for some integer  $l$ . In [13], the authors suggest using  $l = 4$ . In [24], Micciancio suggests  $l = n$ .
- “almost rectangular lattice”: Choose  $\mathbf{R}'$  uniformly at random from  $\{-l, \dots, l\}^{n \times n}$  and add a multiple of the identity matrix  $\mathbf{R} = \mathbf{R}' + k \mathbf{I}_n$ . In [13], the authors suggest  $l = 4, k = \lceil l\sqrt{n} \rceil$ . Micciancio notes in [24]

that this distribution discloses the rough direction of the vectors in  $\mathbf{R}$  to an attacker.

Micciancio showed in [24] that using the Hermite normal form  $\mathbf{B} = \text{HNF}(\mathbf{R})$  as public key is optimal for giving an attacker as little information as possible.

Choose a public threshold parameter  $\tau \in \mathbb{R}$ . Two possibilities are outlined by Goldreich et al. [13]:

- “no signing failure”: Let  $\gamma$  be the maximum  $\ell_1$ -norm of the rows in  $\mathbf{R}$ . Choose  $\tau = \gamma\sqrt{n}/2$ .
- “low-probability signing failure”: Let  $\rho_{\max}$  be the entry in  $\mathbf{R}$  with largest absolute value. In order to guarantee that signing failures have probability less than  $\epsilon$ , choose  $\tau = \rho_{\max} \ln(2n/\epsilon)\sqrt{n}/2$ . In [13], the authors suggest using  $\epsilon = 2^{-30}$ , and claim that heuristically  $\tau$  can be halved in practice and still achieves a probability less than  $\epsilon$  for signing failures.

**Signing.** Hash the message into  $\mathbf{m} \in \mathbb{Z}^n$ . A signature  $\mathbf{s}$  is computed via a CVP approximation algorithm, e.g. Babai’s round-off algorithm<sup>1</sup>

$$\mathbf{s} = \mathbf{R} \lceil \mathbf{R}^{-1} \mathbf{m} \rceil.$$

If signing failures are possible due to a small threshold parameter, we need to check whether  $\|\mathbf{s} - \mathbf{m}\| \leq \tau$ , and fail if this is not the case.

**Verification.** First check that the signature is a lattice vector with the public key  $\mathbf{s} \in \Lambda(\mathbf{B})$  using basic linear algebra. Then check whether  $\|\mathbf{m} - \mathbf{s}\| \leq \tau$ . If both checks pass, the signature is valid.

## 2.2 NTRUSign

In 2003 Hoffstein et al. introduced NTRUSign as a successor to the NTRU signature scheme (NSS) [15], which was broken in 2001 by Gentry et al. [11]. NTRUSign is a special instantiation of GGH, which allows for smaller key and signature sizes and is more efficient. On the other hand, it employs only lattices of a subclass of all lattices related to certain polynomial rings. NTRUSign was under consideration for the IEEE-standard P1363.1 (until draft 7).

---

<sup>1</sup>Alternatively, one could use Babai’s nearest plane algorithm (cf. [3])

There is no proof of security for any variant of NTRUSign, and in 2006 Nguyen and Regev presented an efficient transcript attack on NTRUSign without perturbations [29]. The most recent description of NTRUSign is given by Hoffstein et al. in [14]. We give a brief summary the recommended variant:

**Parameters.** The main security parameter is a prime  $N \in \mathbb{N}$ . Other parameters are modulus  $q \in \mathbb{N}$ , key-size  $d \in \mathbb{N}$ , norm balancing constant  $\beta \in \mathbb{R}$ , acceptance threshold  $\mathcal{N} \in \mathbb{R}$ , and the amount of perturbations  $k \in \mathbb{N}$ .

For 80-bit security, the authors of [14] recommend  $N = 157$ ,  $d = 29$ ,  $q = 256$ ,  $\beta = 0.38407$ ,  $\mathcal{N} = 150.02$ . Their evaluation did not take the 2006 attack by Nguyen and Regev [29] into account.

**Key generation.** Let  $\mathcal{R} = \mathbb{Z}[x]/\langle x^N - 1 \rangle$  be the ring of convolution modular polynomials. Choose uniformly at random two trinary polynomials  $f$  and  $g$  in  $\mathcal{R}$ , such that both have  $d + 1$  coefficients equal to 1,  $d$  coefficients equal to  $-1$ , and the remaining coefficients equal to 0. Restart if either  $f$  or  $g$  is not invertible. Find small polynomials  $F$  and  $G$ , such that  $fG - Fg = q$  over  $\mathcal{R}$  (for the details we refer to [14]).

This process is repeated  $k + 1$  times, so we get  $(k + 1)$  tuples  $(f_0, g_0, F_0, G_0), \dots, (f_k, g_k, F_k, G_k)$ . These are the secret keys. Compute  $h_i \leftarrow f_i^{-1} F_i \bmod q$  for all  $i = 0, \dots, k$ . The public key is  $h \leftarrow h_k$ .

The secret bases corresponding to the  $i$ th secret key is

$$\mathbf{R}_i = \begin{pmatrix} C(f_i) & C(g_i) \\ C(F_i) & C(G_i) \end{pmatrix}, \text{ where } C(a) = \begin{pmatrix} a_0 & a_{N-1} & \cdots & a_1 \\ a_1 & a_0 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1} & a_{N-2} & \cdots & a_0 \end{pmatrix}.$$

The public basis for the lattice  $\Lambda(\mathbf{R}_k)$  is

$$\mathbf{B} = \begin{pmatrix} C(1) & C(0) \\ C(h) & C(q) \end{pmatrix}.$$

The bit-length of the private/public key are approximately

$$l_{\text{pri}} = 2(k + 1) \lceil N \log_2(3) \rceil, l_{\text{pub}} = \lceil N \log_2(q) \rceil.$$

**Signing.** Let  $\mathcal{R}_q$  be  $\mathcal{R}/q\mathcal{R}$ . Hash the message into  $(m_1, m_2) \in \mathcal{R}_q^2$ . Compute  $m \leftarrow m_2 - m_1 h \bmod q$ . The modified message  $(0, m)$  is perturbed by successively approximating CVP in each lattice spanned by a secret basis. For  $i = 0, \dots, k$  compute

$$\begin{pmatrix} s_i \\ t_i \end{pmatrix} = \mathbf{R}_i \left\lceil \mathbf{R}_i \begin{pmatrix} s_{i-1} \\ t_{i-1} \end{pmatrix} \right\rceil, \text{ where } (s_{-1}, t_{-1}) = (0, m)$$

The final  $(s_k, t_k)$  is as close to  $(0, m)$ , as  $(s_k + m_1, t_k + m_1 h \bmod q)$  is to  $(m_1, m_2)$ . The signature is  $s \leftarrow s_k + m_1$ .

The bit-length of a signature is approximately  $l_{\text{sig}} = \lceil N \log_2(q) \rceil$ .

**Verification.** Hash the message again into  $(m_1, m_2) \in \mathcal{R}_q^2$ . Complete the signature  $s$  into the tuple  $(s \bmod q, sh \bmod q)$ . This corresponds to a lattice vector in  $\Lambda(\mathbf{B})$ . We define the following semi-norms on  $\mathcal{R}$  and  $\mathcal{R}_q^2$

$$\begin{aligned} \|\cdot\|_{\mathcal{R}} : \mathcal{R} &\rightarrow \mathbb{R} : \sum_{i=0}^{N-1} a_i x^i \mapsto \sum_{i=0}^{N-1} a_i^2 - \frac{1}{N} \left( \sum_{i=0}^{N-1} a_i \right)^2, \\ \|\cdot\|_{\beta} : \mathcal{R}_q^2 &\rightarrow \mathbb{R} : (a, b) \mapsto \min \left\{ \sqrt{\|a + vq\|_{\mathcal{R}}^2 + \beta^2 \|b + wq\|_{\mathcal{R}}^2} : v, w \in \mathcal{R} \right\} \end{aligned}$$

Check that the distance between message and signature tuple in the  $\beta$ -semi-norm is less than  $\mathcal{N}$ . If it is, the signature is valid.

## 2.3 Attacks and countermeasures

Every message–signature pair in the GGH scheme leaks information about the secret key used to create it. This leak was known to the creators and does not necessarily mean that the scheme is insecure. An attacker might need a transcript of many message–signature pairs in order to mount an attack, or it might even be computationally infeasible to forge a signature with the information leaked by arbitrarily many message–signature pairs.

**Attack.** Nguyen and Regev have shown in 2006 that neither is the case for GGH and NTRUSign without perturbations. They proposed an efficient algorithm to recover the signing key with about 400 signatures for NTRUSign instances with lattices of dimension 502. For GGH, the amount of necessary

signatures is bigger. For example, attacking GGH with lattices of dimension 200 requires about 20,000 signatures. In dimension 400, the amount of necessary signatures is around 160,000. We summarize the attack for two-dimensional lattices in Figure 1).

Let  $\mathbf{m} \in \mathbb{Z}^n$  be the message and  $\mathbf{s} \in \Lambda(\mathbf{R})$  the corresponding signature, which approximates CVP by use of the secret basis  $\mathbf{R}$ . Then

$$\mathbf{s} - \mathbf{m} \in \mathcal{P}_{1/2}(\mathbf{R}) = \{\mathbf{R}\mathbf{x} : \mathbf{x} \in [-1/2, 1/2]^n\}.$$

We will refer to the set  $\mathcal{P}_{1/2}(\mathbf{R})$  as the hidden parallelepiped. We make an experimentally justified assumption at this point that for randomly chosen  $\mathbf{m}$ , the difference  $\mathbf{s} - \mathbf{m}$  is uniformly distributed over  $\mathcal{P}_{1/2}(\mathbf{R})$ . The algorithmic problem is, given enough independent samples from  $U(\mathcal{P}_{1/2}(\mathbf{R}))$ , recover the columns of  $\pm\mathbf{R}$  or an approximation thereof.

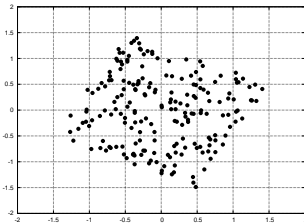


Figure 1: A two-dimensional hidden parallelepiped

See Figure 1 for 200 samples distributed uniformly over a parallelepiped. You can essentially read off the matrix  $\mathbf{R}$ , which describes the parallelepiped, from the graph. This intuition can be made rigorous with algorithms that work efficiently even for high dimensions.

**Countermeasures.** So far, the only known countermeasure against this attack is to use perturbations. When using  $k$  perturbations, the hidden parallelepiped  $\mathcal{P}_{1/2}(\mathbf{R})$  is replaced by the Minkowski sum of the parallelepipeds for each secret basis  $\sum_{i=0}^k \mathcal{P}_{1/2}(\mathbf{R}_i)$ . Only the lattice spanned by the  $k$ th secret basis is public. Though this makes the problem of forging a message harder, it might be possible to adapt the attack we have discussed here to deal with one or more perturbations.

Using perturbations drastically increases key generation and signing time, as well as the size of the secret key. No proof of security is known for either

GGH or NTRUSign with perturbations. The scheme by Gentry, Peikert and Vaikuntanathan discussed in the next section can be seen as a theoretically justified variant of GGH and NTRUSign.

### 3 Signatures based on hash functions

The tree construction of Merkle [23] allows to transform any one-time signature (OTS) scheme into a signature scheme allowing more than one signature. Merkle trees are constructed using a collision resistant hash function and a one-time signature scheme. Thus, if there is a collision resistant hash function and an OTS scheme whose security only relies on the hardness of lattice problems, one could create a lattice based signature scheme.

#### 3.1 Digital signatures using Merkle trees

The Merkle signature scheme (MSS) [23] uses a binary tree for authentication of many one-time signature keys, which makes it possible to create digital signature schemes based on one-time signature schemes.

Merkle trees are constructed as follows: the leaves of the tree are the one-time public keys. Every inner node  $y$  is constructed via

$$y = H(\text{left child} \parallel \text{right child}), \quad (2)$$

where  $H$  is a collision resistant hash function. Using this construction, the values of all inner nodes are predetermined by the leaf values. A tree of height  $h$  can be used to sign  $2^h$  messages. For the verification of a single signature the so-called *authentication path* is required. This path consists of the siblings of the nodes on the path from the bottom to the root of the

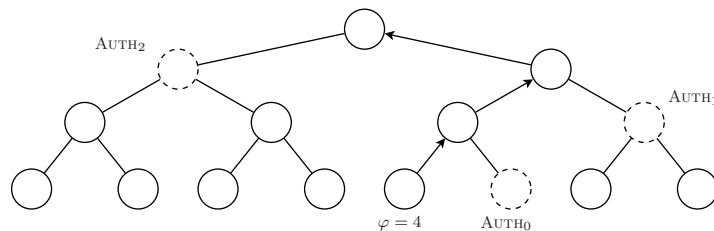


Figure 2: Authentication path of leaf  $\varphi = 4$ .

Merkle tree. The actual authentication node on height  $i$  is labelled  $\text{AUTH}_i$  (see Figure 2 for a demonstration). The Merkle signature scheme is a tuple  $\text{MSS} = (\text{Kg}, \text{Sig}, \text{Vf})$ , where:

**Key generation.**  $\text{MSS.Kg}$  computes all one-time keys and builds the complete Merkle tree to obtain its root. The  $\text{MSS}$  private signing key is the collection of all  $2^h$  one-time signing keys. The root of the tree is published as  $\text{MSS}$  public key.

**Signing.**  $\text{MSS.Sig}$  computes a signature  $\sigma$  consisting of the index  $\varphi$  of the tree leaf, the one-time signature, the one-time verification key, and the authentication path of the current leaf:  $\sigma = (\varphi, \sigma_{OTS}, pk_{OTS}, \{\text{AUTH}_i\}_\varphi)$ .

**Verification.** In a first step, the verification algorithm  $\text{MSS.Vf}$  verifies the one-time signature  $\sigma_{OTS}$  with the key  $pk_{OTS}$ . If it succeeds, it starts the authentication of the one-time public key. Starting from the leaf and using the construction rule (2), the root can be reconstructed using a leaf value and its authentication path.  $\text{MSS.Vf}$  outputs 1 iff the reconstructed root matches the  $\text{MSS}$  public key.

The number of messages that can be signed with  $\text{MSS}$  is restricted to  $2^{20}$ , as with tree heights of  $h > 20$  the key generation needs too much time to be applicable.

**The GMSS extension.** The main disadvantage of the Merkle signature scheme is the size of the signatures. To reduce this size, an extension called *generalized Merkle signature scheme* (GMSS [5]) was proposed in 2007. This construction uses multiple Merkle trees organized in  $T$  layers one upon another. This construction allows to lower the parameters of the underlying one-time signature scheme to obtain faster signature generation time and smaller signatures.

Additionally, using different layers allows to create signatures up to a number of  $2^{80}$ . Using some preprocessing, it is possible to balance the computation of signatures, so that the creation of each signature takes roughly the same time. Considering the timings, GMSS is competing with famous signature schemes like RSA, DSA, or ECDSA, whereas the signature size must be shortened to be comparable to those schemes. For a complete de-

scription we refer to [5]. For the key size analysis we only consider the GMSS extension, as it is the most promising Merkle scheme for practical purposes.

**Authentication path computation.** One important step in the computation of Merkle signatures is the computation of authentication paths. Since subsequent signatures share most of the path from leaf to the root, the siblings on this way are mostly the same, as well. For this reason subsequent signatures share many nodes in the authentication paths. So the challenging problem is to compute the authentication path of successive leaves. This operation is called *authentication path computation*.

There are two different approaches for calculating this authentication path. Merkle proposed a simple method, which uses an algorithm called Treehash. This algorithm computes every inner node  $y$  when needed, using all leafs of the subtree with root  $y$ . Another approach was shown by Jacobsson et al. in [16]. This algorithm uses fractals of the tree to precompute inner nodes. It also allows a trade-off between computation time and storage. Szydło [35, 34] presented a variant of Merkle’s algorithm and showed that his algorithm is asymptotically optimal. He uses  $\mathcal{O}(h)$  computation units and needs  $\mathcal{O}(h)$  storage space to compute authentication paths. In [6], an algorithm is presented by Buchmann, Dahmen, and Schneider that achieves the same asymptotical bounds but balances the signature generation time and lowers the total number of operations. The same paper introduces the possibility of a time-memory trade-off to speed up the computation spending more storage space.

In [5] the Winternitz scheme is used for the one time signatures. We will now present the size of the keys and the signature of GMSS with Winternitz. Let  $\ell$  be the output bitlength of the hash function which is used, and let  $t_w = \lceil \ell/w \rceil + \lceil (\lceil \log_2(\lceil \ell/w \rceil) \rceil + 1 + w)/w \rceil$  be the number of parts of a Winternitz signature. Now  $T$  is the number of tree layers,  $H_i$  is the height of the  $i$ th layer, and  $w_i$  is the corresponding Winternitz parameter. Let  $m_{\text{auth}}(i) = (3H_i + \lfloor H_i/2 \rfloor - 3K - 2 + 2^K) \cdot \ell$  bits be the storage required for one authentication path together with the algorithm state.

The GMSS public key consists of one hash value, namely the tree root (therefore  $m_{\text{pubkey}} = \ell$  bits). The private key requires

$$m_{\text{privkey}} = \left( \sum_{i=1}^T (m_{\text{auth}}(i) + 1) + \sum_{i=2}^T (m_{\text{auth}}(i) + t_{w_{i-1}} + 2) \right) \cdot \ell \text{ bits}.$$

A GMSS signature  $\sigma$  consists of  $m_{\text{signature}} = \sum_{i=1}^T (H_i + t_{w_i}) \cdot \ell$  bits. Refer to [4] for the complete analysis of these numbers.

### 3.2 Lattice-based hash functions

For construction of the Merkle tree and the Winternitz OTS scheme, any collision resistant hash function can be applied. The basis of lattice-based hash functions is a proposal of Ajtai, who defines a lattice-based, keyed hash function

$$f_{\mathbf{A}} : \{0, \dots, d-1\}^m \rightarrow \mathbb{Z}_q^n \quad f_{\mathbf{A}}(\mathbf{y}) = \mathbf{A}\mathbf{y} \pmod{q}$$

for parameters  $q, n, m, d \geq 1$  and a key matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ . The parameter  $n$  determines the level of security. The function is proven to be one-way, if  $m$  meets the condition  $m > n \log(q) / \log(d)$ . Otherwise the domain of  $f_{\mathbf{A}}$  would be smaller than its range.

Furthermore, the function key  $\mathbf{A}$  consists of  $\mathcal{O}(n^2 \log(n))$  entries in  $\mathbb{Z}_q$  and, to get a reasonable amount of security, the computation of  $f_{\mathbf{A}}$  requires too many arithmetic operations. To avoid this disadvantage, Micciancio proposed the use of cyclic lattices [25]. With this construction only  $\mathcal{O}(n \log(n))$  entries have to be stored for the function key and the computation time can be shortened using the Fast Fourier Transform (FFT) to  $\tilde{\mathcal{O}}(n)$ . But this function lacks collision resistance.

A collision resistant hash function based on the worst case hardness of SVP was introduced independently by Peikert and Rosen [30] and by Micciancio and Lyubashevsky [20]. They used ideal lattices to construct a keyed hash function. The hash computation is the same as above:

$$f_{\mathbf{A}} : \{0, \dots, d-1\}^m \rightarrow \mathbb{Z}_q^n \quad f_{\mathbf{A}}(\mathbf{y}) = \tilde{\mathbf{A}}\mathbf{y} \pmod{q},$$

where  $\tilde{\mathbf{A}} = [\mathbf{a}_1 \mathbf{F}\mathbf{a}_1 \mathbf{F}^2\mathbf{a}_1 \dots \mathbf{F}^{n-1}\mathbf{a}_1 | \mathbf{a}_2 \mathbf{F}\mathbf{a}_2 \dots \mathbf{F}^{n-1}\mathbf{a}_2 | \dots | \mathbf{a}_{m/n} \dots \mathbf{F}^{n-1}\mathbf{a}_{m/n}]$  for  $\mathbf{a}_1 \dots \mathbf{a}_{m/n} \in \mathbb{Z}_q^n$  and

$$\mathbf{F} = \left[ \begin{array}{c|c} 0 \cdots 0 & -\mathbf{v} \\ \hline I_{n-1} & \end{array} \right].$$

Here the entries of  $\mathbf{v}$  are the coefficients of the polynomial  $v$  that defines the ideal lattice, namely  $v(x) = v_0 + v_1x + \dots + v_{n-1}x^{n-1} + x^n$ .

Suppose  $v$  to be irreducible over  $\mathbb{Z}$ , and for any polynomial  $g$  suppose the norm  $\|g \pmod{v}\|_{\infty}$  not to be much bigger than  $\|g\|_{\infty}$  (compare the full

version of [20]). Then using a worst case to average case reduction, similar to the idea of Ajtai [1], it is possible to prove the following: if an adversary is able to find a collision in the randomly chosen hash function then there exists a polynomial time algorithm that solves the approximate shortest vector problem in the worst case for all ideal lattices of some polynomial ring with polynomial approximation factor [20].

SWIFFT [22], a variant of these lattice based hash functions, is even competing with the SHA-2 hash family. While the computation timings are comparable for both families, the lattice-based hash functions, i.e. the SWIFFT family, are even provably secure under feasible assumptions (namely the hardness of approximating SVP in ideal lattices up to polynomial factors). The output of SWIFFT is a vector in  $\mathbb{Z}_q^n$ , so the output length is in  $\mathcal{O}(n \log(n))$ .

### 3.3 One-time signatures based on SVP-hardness

Using the Winternitz OTS scheme, it is possible to turn any collision resistant hash function into a one-time signature scheme. This construction leads to a first OTS scheme based on the hardness of lattice problems. The main disadvantage of the Winternitz scheme is the signature size, which is  $\mathcal{O}(\ell^2)$ .

In [21], Lyubashevsky and Micciancio present a one-time signature scheme based on the hardness of finding short vectors in ideal lattices, with signature size  $\mathcal{O}(n \log(n))$ . Therefore using the Lyubashevsky/Micciancio one-time scheme and a lattice based hash function one could create a signature scheme based on lattices. This section introduces the mentioned OTS scheme of [21].

Let  $R = \mathbb{Z}_p[x]/\langle v(x) \rangle$  a ring with an irreducible polynomial  $v \in \mathbb{Z}[x]$  of degree  $n$ . Define a ring constant

$$\phi(R) = \min_{j \in \mathbb{N}} \{ \forall \mathbf{a}, \mathbf{b} \in R : \|\mathbf{a}\mathbf{b}\|_\infty \leq jn \|\mathbf{a}\|_\infty \|\mathbf{b}\|_\infty \}$$

(e.g.  $R = \mathbb{Z}_p[x]/\langle x^n + 1 \rangle \Rightarrow \phi(R) = 1$ , see [21]).

**Key generation.** Let  $p = (\phi n)^3$ ,  $m = \lceil \log(n) \rceil$ ,  $R$  as above. For the signing key, choose  $(\mathbf{k}, \mathbf{l}) \in R^m \times R^m$  with

$$\|\mathbf{k}\|_\infty \leq 5jp^{1/m} \quad \text{and} \quad \|\mathbf{l}\|_\infty \leq 5jn\phi p^{1/m}$$

with  $j$  chosen as described in [21]. The verification key is  $(\mathbf{H}, \mathbf{H}(\mathbf{k}), \mathbf{H}(\mathbf{l}))$ , where  $\mathbf{H}$  is an element of a hash function family  $\mathcal{H}_{R,m}$  that maps elements from  $R^m$  to  $R$ .

**Signing.** On input a message  $M \in R$  with  $\|M\|_\infty \leq 1$  the signing process Sig performs the following:  $\sigma \leftarrow \mathbf{k}M + \mathbf{1}$ .

**Verification.** Accept iff  $\|\sigma\|_\infty \leq 10\phi p^{1/m}n \log^2(n)$ ,  $\mathbf{H}(\sigma) = \mathbf{H}(\mathbf{k})M + \mathbf{H}(\mathbf{1})$ .

It can be shown that this one-time signature scheme is  $(t, 1, q_{\mathbf{H}}, \epsilon)$ -strongly unforgeable under chosen message attacks (SU-CMA, where the number of calls to the signature oracle has to be bounded by 1). If there is a successful polynomial time adversary against strong unforgeability, this adversary can be used to find short vectors in all lattices that correspond to the ideals in the ring  $R$  [21].

Replacing the Winternitz scheme by the OTS scheme of Lyubashevsky and Micciancio leads to the following key and signature sizes. The size of the public key remains the same as before, it still consists of one hash value. The size of a LM OTS signature is the size of  $s \in R^m$  where  $m = \lceil \log(n) \rceil$ . An element of  $R$  has size less than  $3 \lceil \log(n) \rceil$ , so the signature has size of less than  $3n \lceil \log(n) \rceil = \mathcal{O}(n \log(n))$ . The GMSS private key requires to store a bit amount of

$$m_{\text{privkey}} = \left( \sum_{i=1}^T (m_{\text{auth}}(i) + 1) + \sum_{i=2}^T (m_{\text{auth}}(i) + 2) \right) \cdot \ell + 3(T-1)n \lceil \log(n) \rceil ,$$

where the first part is the storage needed for the hash tree and the second part is the size of the required one-time signatures. For a signature one needs to store  $m_{\text{signature}} = \sum_{i=1}^T (H_i) \cdot \ell + 3Tn \lceil \log(n) \rceil$  bits.

## 4 Signatures based on presampling

Following the ideas of the GGH and NTRU signature schemes as well as recent observations on Gaussians over lattices by Micciancio and Regev, Gentry, Peikert, and Vaikuntanathan (GPV) proposed an efficient signature scheme that is provably secure in the random oracle model [12]. It uses Klein's modification of the Babai nearest plane algorithm [3, 18] and two famous results by Ajtai [1, 2] in order to build a trapdoor function with preimage sampling that almost behaves like a trapdoor permutation. The basis of security is the collision resistance of the proposed trapdoor function, which in turn is based on the worst-case hardness of finding short vectors in *general*, esp. not ideal, lattices. We refer the reader to [12] and to [27] for

further details and a comprehensive discussion of the involved lattice problems and on Gaussians in the lattice context. The practical hardness of these lattice problems is analyzed in [7] and subsequently in [8].

## 4.1 Lattice-based trapdoor function

In the following, we specify a family of trapdoor functions and discuss its realization based on [12]. They use modular lattices and a Gaussian sampling algorithm (Algorithm A.1) based on Babai’s nearest plane algorithm [3]. We will refer to the following construction by **GPV trapdoor**. The family is described via a triple  $(\text{TrapGen}, \text{SampleDom}, \text{SamplePre})$ , where:

**Trapdoor Generation.**  $\text{TrapGen}(n)$  chooses a prime modulus  $q \approx n^3$ , the domain dimension  $m = 5n \log(q)$ , the basis length bound  $L = m^{1+\epsilon}$ ,  $\epsilon > 0$ , and a Gaussian parameter  $s = L \omega(\sqrt{\log(m)})$ .

The algorithm sets up the range as  $R_n = \mathbb{Z}_q^n$  and the domain  $D_n$  as

$$D_n = \{\mathbf{e} \in \mathbb{Z}^m : \|\mathbf{e}\|_2 \leq s \sqrt{m}\}.$$

The public trapdoor key  $a$  describes the above public parameters and the public matrix  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ , which is chosen randomly along with a set  $\mathbf{S}$  of short binary vectors as in [2]. The set

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \mathbf{A} \mathbf{v} \equiv \mathbf{0} \pmod{q}\}$$

describes a lattice, for which the secret trapdoor parameter  $t$  describes a basis  $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_m]$  such that for its Gram-Schmidt orthogonalization  $\tilde{\mathbf{T}}$

$$\max_{i=1, \dots, m} \|\tilde{\mathbf{t}}_i\|_2 \leq L.$$

A basis with this property can be constructed by applying [26, Lemma 7.1] on  $\mathbf{S}$ . Finally, the output of **TrapGen** is the public key  $a = (n, q, m, L, s, \mathbf{A})$  and the secret key  $t = \mathbf{T}$ .

**Trapdoor evaluation.** On input  $\mathbf{x}$ , the trapdoor function  $f_a(\mathbf{x})$  evaluates  $\mathbf{y} \leftarrow \mathbf{A} \mathbf{x} \pmod{q}$  and returns  $\mathbf{y}$ .

**Domain Sampling (SampleDom).** Sampling from  $D_n$  is performed via Algorithm A.1 using the standard basis of  $\mathbb{Z}^m$ .

**Preimage Sampling (SamplePre).** Sampling from  $f_t^{-1}$  is performed via Algorithm A.1 using  $\mathbf{T}$ . On input  $\mathbf{y}$ ,  $\text{SamplePre}(t, \mathbf{y})$  performs the following steps:

1. Compute  $\mathbf{t} \in \mathbb{Z}_q^m$ , such that  $\mathbf{A} \mathbf{t} \equiv \mathbf{y} \pmod{q}$ . This is done by linear algebra and almost certainly yields a  $\mathbf{t} \notin D_n$ .
2. Use the trapdoor basis  $\mathbf{T}$  in Algorithm A.1 to sample a vector  $\mathbf{v}$  from a Gaussian distribution around  $-\mathbf{t}$  with standard deviation  $s$ , conditioned with  $\mathbf{A} \mathbf{v} \equiv \mathbf{0} \pmod{q}$ , and output  $\mathbf{x} = \mathbf{t} + \mathbf{v} \in D_n$ .

We argue that the GPV trapdoor satisfies the following important properties.

**Function generation.** There is an efficient algorithm  $\text{TrapGen}$  that outputs  $(a, t) \leftarrow \text{TrapGen}(n)$ , where  $a$  fully defines the function  $f_a$  and the trapdoor  $t$  is used to sample from the inverse  $f_t^{-1}(\cdot)$ , which is defined as  $\text{SamplePre}(t, \cdot)$ . More precisely, the function generation can be performed in polynomial time in the security parameter by applying results from [2, 12, 26].

**Efficiency.** The function  $f_a : D_n \rightarrow R_n$  is efficiently computable. Furthermore, the two finite sets  $R_n, D_n$  are efficiently recognizable and  $R_n$  is closed under addition. Since it is a simple modular matrix-vector multiplication, trapdoor evaluation is very efficient and the sets  $D_n, R_n$  can obviously be efficiently recognized by performing a simple norm check.

**One-wayness.** Computing the function  $f_t^{-1} : R_n \rightarrow D_n$ , is infeasible without the trapdoor  $t$  under the assumption that solving  $\text{ISIS}(q, m, s \sqrt{m})$  is hard on the average.

**Domain sampling with uniform output.**  $\text{SampleDom}(n)$  samples from some distribution over  $D_n$ , such that their images under  $f_a$  are uniformly distributed over  $R_n$  (cf. [12]).

**Preimage sampling.** Let  $\mathbf{y} \in R_n$ .  $f_t^{-1}(\mathbf{y})$  samples  $\mathbf{x} \leftarrow \text{SampleDom}(n)$  under the condition that  $f_a(\mathbf{x}) = \mathbf{y}$ . The entropy of  $\mathbf{x}$  is at least  $\omega(\log(n))$  (cf. [12]). Explicitly, we have

$$|\{\mathbf{x} \leftarrow \text{SampleDom}(n) : f_a(\mathbf{x}) = \mathbf{y}\}| \geq 2^{\omega(\log(n))}.$$

**Linearity.**  $f_a$  is linear in the sense that for all  $\mathbf{x}_1 + \mathbf{x}_2 \in D_n$

$$f_a(\mathbf{x}_1 + \mathbf{x}_2) \equiv \mathbf{A}(\mathbf{x}_1 + \mathbf{x}_2) \equiv \mathbf{A}\mathbf{x}_1 + \mathbf{A}\mathbf{x}_2 \equiv f_a(\mathbf{x}_1) + f_a(\mathbf{x}_2) \pmod{q}.$$

**Collision resistance.** There is no algorithm  $\mathcal{A}(n, a)$  that outputs a pair  $(\mathbf{x}, \mathbf{x}') \in D_n^2$ , such that  $\mathbf{x} \neq \mathbf{x}'$  and  $f_a(\mathbf{x}) = f_a(\mathbf{x}')$ , in time polynomial in  $n$  with noticeable probability. This holds under the assumption that  $\text{SIS}(q, m, 2s\sqrt{m})$  is hard on the average. The proof is straightforward. Simply compute  $\mathbf{z} \leftarrow \mathbf{x} - \mathbf{x}'$  and observe that  $\mathbf{z} \in \Lambda_q^\perp(\mathbf{A})$  and  $\|\mathbf{z}\|_2 \leq s\sqrt{m} + s\sqrt{m}$ , which solves the SIS. As in the original work, we will always assume that the above properties, especially the statistical distributions, hold in a perfect sense. For efficiency reasons, the parameters  $n, q, m, L, s$  can be shared among all signers and can be considered fixed system parameters.

## 4.2 GPV signature scheme

Using the above trapdoor, the GPV signature scheme is a tuple  $(\text{Kg}, \text{Sig}, \text{Vf})$ :

**Key generation.**  $\text{GPV.Kg}(n)$  outputs  $(a, t) \leftarrow \text{TrapGen}(n)$ . Furthermore, the algorithm chooses a collision resistant full domain hash function  $\text{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q^m$  from the family of all such functions.

**Signing.** Let  $M \in \{0, 1\}^*$  be a message.  $\text{GPV.Sig}(t, M)$  checks whether  $M$  has been signed before. If so, it outputs the same signature. Otherwise, it outputs  $\sigma \leftarrow \text{SamplePre}(t, \text{H}(M))$ ,  $\sigma \in D_n$ , and stores  $(M, \sigma)$ .

**Verification.** On input  $(M, \sigma)$ ,  $\text{GPV.Vf}(a, \sigma, M)$  returns 1 iff  $\sigma \in D_n$  and  $f_a(\sigma) = \text{H}(M)$ .

Observe that GPV is complete because for all honestly generated keys  $(a, t) \leftarrow \text{TrapGen}(n)$ , all  $M \in \{0, 1\}^*$ , and all  $\sigma \leftarrow \text{GPV.Sig}(M)$ , we have

$$f_a(\sigma) \equiv f_a(\text{SamplePre}(\text{H}(M))) \equiv f_a(f_t^{-1}(\text{H}(M))) \equiv \text{H}(M) \pmod{q}$$

and therefore  $\text{GPV.Vf}(a, \sigma, M) = 1$ .

Note that the signature scheme is stateful, which makes it less applicable in some scenarios. There is, however, a stateless modification  $\text{GPV}'$  in [12] that uses standard randomization techniques. The modified signature and verification procedures are as follows:

**Signing.**  $\text{GPV}'.\text{Sig}(t, M)$  chooses  $r \leftarrow \{0, 1\}^k, k = \omega(\log(n))$ , uniformly at random and executes  $\sigma \leftarrow \text{GPV}.\text{Sig}(t, M||r)$ . The output is  $(r, \sigma)$

**Verification.** Given a signature  $\sigma$ ,  $\text{GPV}'.\text{Vf}(a, (r, \sigma), M)$  returns 1 iff  $\sigma \in D_n$  and  $f_a(\sigma) = \text{H}(M||r)$ . Due to the random choice of  $r$  from a large set, collisions occur only with negligible probability.

Like GPV, the modified GPV' is strongly-unforgeable in the random oracle model. A security proof for GPV' can be found in Section 4.3. It is similar to the proof for GPV in [12]. From the given parameters for GPV, we can deduce an approximate signature size of  $22.5 n \log_2(n) \log_2(15 n \log_2(n))$  bits as well as the size of public and private keys  $15 n^2 \log_2^2(n)$ .

### 4.3 Security proof

As stated before, both GPV and GPV' are provably secure under an average-case hardness assumption on the SIS. As for GPV, we simply repeat the precise version of the respective theorem and refer the interested reader to [12] for the proof. In the following let  $T_{\text{TrapGen}}(n)$ ,  $T_{\text{SampleDom}}(n)$ ,  $T_{f_a}(n)$ , and  $T_{\text{list}}(n)$  be the cost functions for key generation, domain sampling, trapdoor evaluation, and list processing w.r.t. the security parameter.

**Theorem 4.1** (Strong unforgeability of GPV). *Let  $n$  be the security parameter and let  $q(n), m(n), s(n)$  be as above. The GPV signature scheme is  $(t, q_{\text{Sig}}, q_{\text{H}}, \epsilon)$ -strongly-unforgeable if  $\text{H}$  is collision resistant and the  $\text{SIS}(q, m, 2s\sqrt{m})$  is  $(t', \epsilon')$ -hard with*

$$t' = t + q_{\text{Sig}} T_{\text{list}}(n) + q_{\text{H}} (T_{\text{SampleDom}}(n) + T_{f_a}(n) + T_{\text{list}}(n)) \text{ and } \epsilon' = \epsilon - \frac{1}{2^{\omega(\log(n))}}.$$

*Proof.* See [12]. □

As for the stateless variant GPV', the authors of [12] sketch a security proof. The precise theorem can be stated as follows. Observe that the proof is not as tight as for GPV.

**Theorem 4.2** (Strong unforgeability of GPV'). *Let  $n$  be the security parameter and let  $q(n), m(n), s(n)$  be as above. GPV' is  $(t, q_{\text{Sig}}, q_{\text{H}}, \epsilon)$ -strongly unforgeable if  $\text{H}$  is collision resistant and  $\text{SIS}(q, m, 2s\sqrt{m})$  is  $(t', \epsilon')$ -hard with*

$$t' = t + (q_{\text{Sig}} + q_{\text{H}}) (T_{\text{SampleDom}}(n) + T_{f_a}(n) + T_{\text{list}}(n)) \text{ and } \epsilon' = \epsilon - \frac{q_{\text{Sig}}^2 + 1}{2^{\omega(\log(n))}}.$$

*Proof.* See Appendix B □

## References

- [1] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1996*, pages 99–108. ACM Press, 1996.
- [2] Miklós Ajtai. Generating hard instances of the short basis problem. In *International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, pages 1–9. Springer-Verlag, 1999.
- [3] László Babai. On Lovász’ lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.
- [4] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post-Quantum Cryptography*. Springer-Verlag, 2008.
- [5] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *International Conference on Applied Cryptography and Network Security — ACNS 2007*, Lecture Notes in Computer Science, pages 31–45. Springer-Verlag, 2007.
- [6] Johannes Buchmann, Erik Dahmen, and Michael Schneider. Merkle tree traversal revisited. In *Post-Quantum Cryptography (PQCrypto) 2008*, Lecture Notes in Computer Science, pages 63–78. Springer-Verlag, 2008.
- [7] Johannes Buchmann, Richard Lindner, and Markus Rückert. Explicit hard instances of the shortest vector problem. In *Post-Quantum Cryptography (PQCrypto) 2008*, Lecture Notes in Computer Science, pages 79–94. Springer-Verlag, 2008.
- [8] Johannes Buchmann, Richard Lindner, Markus Rückert, and Michael Schneider. Explicit hard instances of the shortest vector problem (extended version). Number 2008/333 in Cryptology ePrint Archive. [eprint.iacr.org](http://eprint.iacr.org), 2008.
- [9] Irit Dinur. Approximating  $SVP_\infty$  to within almost-polynomial factors is NP-hard. *Theoretical Computer Science*, (1):55–71, 2002.

- [10] Nicolas Gama, Nick Howgrave-Graham, and Phong Q. Nguyen. Symplectic lattice reduction and ntru. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 233–253. Springer, 2006.
- [11] Craig Gentry, Jakob Jonsson, Jacques Stern, and Michael Szydlo. Cryptanalysis of the ntru signature scheme (nss) from eurocrypt 2001. In *Advances in Cryptology — Asiacrypt 2001*, pages 1–20, 2001.
- [12] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2008*, pages 197–206. ACM Press, 2008.
- [13] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology — Crypto 1997*, Lecture Notes in Computer Science, pages 112–131. Springer-Verlag, 1997.
- [14] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Performance improvements and a baseline parameter generation algorithm for NTRUsign. <http://grouper.ieee.org/groups/1363/lattPK/submissions.html>, 2005.
- [15] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An ntru lattice-based signature scheme. In *Advances in Cryptology — Eurocrypt 2001*, Lecture Notes in Computer Science, pages 211–228. Springer-Verlag, 2001.
- [16] Markus Jakobsson, T. Leighton, Silvio Micali, and M. Szydlo. Fractal merkle tree representation and traversal. In *Topics in Cryptology — Cryptographer’s Track, RSA Conference (CT-RSA) 2003*, Lecture Notes in Computer Science, pages 314–326. Springer-Verlag, 2003.
- [17] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, (5):789–808, 2005.
- [18] Philip N. Klein. Finding the closest lattice vector when it’s unusually close. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA) 2000*, pages 937–941. ACM Press, 2000.

- [19] Arjen Lenstra, Hendrik Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, (4):515–534, 1982.
- [20] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *International Colloquium on Automata, Languages and Programming (ICALP) 2006*, Lecture Notes in Computer Science, pages 144–155. Springer-Verlag, 2006.
- [21] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *Theory of Cryptography Conference (TCC) 2008*, Lecture Notes in Computer Science, pages 37–54. Springer-Verlag, 2008.
- [22] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swift: A modest proposal for fft hashing. In *Fast Software Encryption (FSE) 2008*, Lecture Notes in Computer Science, pages 54–72. Springer-Verlag, 2008.
- [23] Ralph Merkle. A certified digital signature. In *Advances in Cryptology — Crypto 1989*, Lecture Notes in Computer Science, pages 218–238. Springer-Verlag, 1990.
- [24] Daniele Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and Lattices (CaLC) 2001*, Lecture Notes in Computer Science, pages 126–145. Springer-Verlag, 2001.
- [25] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 356–365, 2002.
- [26] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.
- [27] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

- [28] Phong Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto'97. In *Advances in Cryptology — Crypto 1999*, Lecture Notes in Computer Science, pages 288–304. Springer-Verlag, 1999.
- [29] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In *Advances in Cryptology — Eurocrypt 2006*, Lecture Notes in Computer Science, pages 215–233. Springer-Verlag, 2006.
- [30] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography Conference (TCC) 2006*, Lecture Notes in Computer Science, pages 145–166. Springer-Verlag, 2006.
- [31] Oded Regev. On the complexity of lattice problems with polynomial approximation factors, 2007. A survey for the LLL+25 conference.
- [32] Oded Regev and Ricky Rosen. Lattice problems and norm embeddings. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2006*, pages 447–456. ACM Press, 2006.
- [33] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [34] Michael Szydło. Merkle tree traversal in log space and time (preprint version), 2003. Available at <http://www.szydlo.com>.
- [35] Michael Szydło. Merkle tree traversal in log space and time. In *Advances in Cryptology — Eurocrypt 2004*, Lecture Notes in Computer Science, pages 541–554. Springer-Verlag, 2004.

## A Randomized Babai Algorithm

**Input.** An  $m$ -dimensional basis  $\mathbf{B}$  of a lattice  $\Lambda$ , a parameter  $s > 0$ , and center vector  $\mathbf{c} \in \mathbb{R}^m$

1.  $\mathbf{v}_m \leftarrow \mathbf{0}$
2.  $\mathbf{c}_m \leftarrow \mathbf{c}$
3. Let  $\tilde{\mathbf{B}}$  be the Gram-Schmidt orthogonalized  $\mathbf{B}$ .
4. For  $i = m, \dots, 1$  do
  - (a)  $c'_i \leftarrow \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle / \langle \mathbf{c}_i, \tilde{\mathbf{b}}_i \rangle$
  - (b)  $s'_i \leftarrow s / \|\tilde{\mathbf{b}}_i\|_2$
  - (c) Choose  $z_i$  from the Gaussian distribution  $D_{\mathbb{Z}, s'_i, c'_i}$ ,  
i.e.  $z_i \in \mathbb{Z} \cap [c'_i - s'_i, c'_i + s'_i]$  with high probability.
  - (d)  $\mathbf{c}_{i-1} \leftarrow \mathbf{c}_i - z_i \tilde{\mathbf{b}}_i$
  - (e)  $\mathbf{v}_{i-1} \leftarrow \mathbf{v}_i - z_i \tilde{\mathbf{b}}_i$

**Output.** The vector  $\mathbf{v}_0$ , such that  $\|\mathbf{v} - \mathbf{c}\|_2 \leq s\sqrt{m}$  with high probability.

**Algorithm A.1:** Modified Babai's nearest plane algorithm. The algorithm samples from a discrete Gaussian distribution over the lattice  $\Lambda$  around the center  $\mathbf{c}$  with standard deviation  $s$ .

## B Proof of Theorem 4.2

*Proof.* Towards contradiction, we assume that there exists a successful forger  $\mathcal{A}$  against strong unforgeability of  $\text{GPV}'$  in the  $\text{Exp}_{\mathcal{A}, \text{GPV}'}^{\text{su-cma}}$  experiment. Using  $\mathcal{A}$ , we construct an algorithm  $\mathcal{B}$  via a black-box simulation, such that  $\mathcal{B}$  solves the  $\text{SIS}(q, m, 2s\sqrt{m})$  with probability  $\epsilon'$  in time  $t'$ . The following simulation uses random oracle techniques to simulate the signature oracle.

**Setup.**  $\mathcal{B}$  gets as input the public trapdoor parameter  $a$  and implicitly the lattice  $\Lambda_q^\perp(A)$ . The algorithm initializes a list  $L_{\mathbf{H}} \leftarrow \emptyset$  of tuples  $(x, r, h, \sigma)$ , indexed by  $x$  in order to simulate  $\mathbf{H}$  and  $f_a$  consistently. It runs  $\mathcal{A}$  on input  $a$  in a black-box simulation.

**Random oracle queries.** On input  $x = M||r$ ,  $r \in \{0, 1\}^n$ ,  $\mathcal{B}$  looks up  $x$  in  $L_{\mathcal{H}}$ . If it finds a tuple  $(x, r, h, *)$  then it returns  $h$ . Otherwise,  $\mathcal{B}$  chooses a  $\sigma \leftarrow \text{SampleDom}(n)$  and computes  $h \leftarrow f_a(\sigma)$ , and stores  $(x, r, h, \sigma)$  in  $L_{\mathcal{H}}$ . It returns  $h$ . In case  $|x| < k$ ,  $\mathcal{B}$  looks up a tuple  $(x, *, h, *)$  in  $L_{\mathcal{H}}$ . If it exists, the algorithm returns  $h$ . Otherwise,  $\mathcal{B}$  chooses a  $\sigma \leftarrow \text{SampleDom}(n)$ , computes  $h \leftarrow f_a(\sigma)$ , stores  $(x, \square, h, \square)$ , and returns  $h$ .

**Signature Queries.** On input  $M \in \{0, 1\}^*$ , algorithm  $\mathcal{B}$  chooses  $r \leftarrow \{0, 1\}^k$  uniformly at random and runs  $\mathcal{H}(M||r)$ , yielding a tuple  $(x, r, h, \sigma) \in L_{\mathcal{H}}$ . The simulator returns  $(r, \sigma)$ .

**Output.** Finally,  $\mathcal{A}$  stops and returns a valid forgery  $(M^*, r^*, \sigma^*)$  with  $\sigma^* \in D_n$  and  $f_a(\sigma^*) \equiv h^* \equiv \mathcal{H}(M^*||r^*) \pmod{q}$ . W.l.o.g., there is a tuple  $(M^*||r^*, r^*, h^*, \sigma) \in L_{\mathcal{H}}$  with  $\sigma \in D_n$ . Algorithm  $\mathcal{B}$  outputs  $\sigma^* - \sigma$  as its solution to SIS.

**Analysis.** Observe that  $\mathcal{B}$  simulates the random oracle and the signature oracle perfectly and consistently. The choice of  $h$  in the simulation of the random oracle is uniform at random because of the uniform output distribution of  $f_a$ . Note that in case  $\mathcal{A}$  queries the random oracle with messages of length less than  $k$ , we only need to store its hash value and not its signature because such messages will never be signed by the signature oracle.

As for the output of  $\mathcal{B}$ , we have to show that  $\sigma^* - \sigma \neq \mathbf{0}$  holds but with negligible probability. We have to distinguish three cases:

1. The adversary  $\mathcal{A}$  outputs a forgery in the strong sense, i.e. he has previously queried his signature oracle on  $M^*$ . Then, we have  $\sigma^* - \sigma \neq \mathbf{0}$  by definition. Note that, due to the collision resistance of  $\mathcal{H}$ , the adversary  $\mathcal{A}$  cannot find a second message for a known signature.
2. Algorithm  $\mathcal{A}$  has not queried his signature oracle on  $M^*$  and  $\mathcal{B}$  has never returned  $\sigma^*$  in any query. W.l.o.g., he has queried  $\mathcal{H}$  on  $M^*$  and  $\mathcal{B}$  has a triple  $(M^*||r^*, r^*, h^*, \sigma) \in L_{\mathcal{H}}$ . By the minimum conditional entropy  $\omega(\log(n))$  of  $\sigma$  (see Section 4.1), we infer that  $\sigma^* = \sigma$  with probability at most  $2^{-\omega(\log(n))}$ , which is negligible.
3. Algorithm  $\mathcal{A}$  has not queried his signature oracle on  $M^*$  but  $\mathcal{B}$  has returned  $\sigma^*$  as a signature on a different message. This only happens with

negligible probability because of the random choice of  $r$  from a large set and the fact that  $\mathcal{A}$  only makes polynomially many queries. Thus,  $\mathcal{B}$  never signs the same value  $\mathbf{H}(M||r)$  but with a negligible probability of at most  $q_{\text{Sig}}^2/2^{\omega(\log(n))}$ .

Since  $\sigma^*$  and  $\sigma$  are valid signatures on  $m^*$ , we have

$$f_a(\sigma^*) = \mathbf{H}(m^*) = f_a(\sigma)$$

and therefore a non-zero vector  $\sigma^* - \sigma \in \Lambda_q^\perp(A)$  of norm  $\|\sigma^* - \sigma\|_2 \leq 2s\sqrt{m}$ .

The overhead of the reduction is dominated by the computational cost for domain sampling and trapdoor evaluation in the simulation of the random oracle with some additional list processing. In the worst-case, the adversary  $\mathcal{A}$  never queries  $\mathbf{H}$  and  $\text{Sig}$  with the same message, which is why the overhead is  $(q_{\text{Sig}} + q_{\mathbf{H}})(T_{\text{SampleDom}}(n) + T_{f_a}(n) + T_{\text{list}}(n))$ .  $\square$