Improvement and Efficient Implementation of a Lattice-based Signature Scheme

Rachid El Bansarkhani and Johannes Buchmann

Technische Universität Darmstadt
Fachbereich Informatik
Kryptographie und Computeralgebra,
Hochschulstraße 10, 64289 Darmstadt, Germany
{elbansarkhani,buchmann}@cdc.informatik.tu-darmstadt.de

Abstract. Lattice-based signature schemes constitute an interesting alternative to RSA and discrete logarithm based systems which may become insecure in the future, for example due to the possibility of quantum attacks. A particularly interesting scheme in this context is the GPV signature scheme [GPV08] combined with the trapdoor construction from Micciancio and Peikert [MP12] as it admits strong security proofs and is believed to be very efficient in practice. This paper confirms this belief and shows how to improve the GPV scheme in terms of space and running time and presents an implementation of the optimized scheme. A ring variant of this scheme is also introduced which leads to a more efficient construction. Experimental results show that GPV with the new trapdoor construction is competitive to the signature schemes that are currently used in practice.

Keywords: Lattice-Based Cryptography, Practicality, Implementations

1 Introduction

The security notion of most cryptographic applications changes in the presence of quantum computers. In the breakthrough work [Sho97] in 1994, Shor pointed out that cryptographic schemes with security based on the hardness of number theoretic assumptions can efficiently be attacked by quantum computers. Since then, many efforts have been spent on the search for alternatives in order to face this challenge. Lattice-based cryptography is a promising candidate that has the potential to meet the security needs of future cryptographic applications. This is mainly due to Ajtai's worst-case to average-case reductions [Ajt96], which attracted a lot of researchers into the field of lattice-based cryptography. Specifically, it states that attacking random instances of a cryptosystem is at least as hard as solving all instances of the underlying lattice problem. As opposed to the discrete log problem and factoring, lattice problems are conjectured to withstand quantum attacks. In the last couples of years, a number of efficient cryptosystems emerged that base the security on the hardness of well-studied lattice problems. Unlike number theoretic constructions such as RSA, there exists no subexponential time attack on lattice problems up to date. All known attacks have exponential time complexity and thus serve as a further supporting argument for a replacement by lattice-based cryptosystems. Based on this observation, one realizes an inherent need to develop new cryptographic primitives that can be based on worst-case lattice problems.

1.1 Our Contribution

In this paper we give the first software implementation of the GPV signature [GPV08] scheme using the newest trapdoor construction from Micciancio and Peikert [MP12]. Moreover, we present an efficient ring variant of the scheme based on the ring-LWE problem. In addition, we propose improvements that lower the memory claims for the perturbation matrix by a factor of about 240 compared to the proposal in [MP12]. When generating signatures the perturbation matrix is required to sample integer vectors with a given covariance. In both variants the matrix and ring variant we considerably improved the running times of key and signature generation. For instance, the running times of key and signature generation are lowered by a factor of 30-190 respectively 2-6 in the ring variant. By providing running times, storage sizes and security levels for different parameter sets we show that the ring variant has a 3-6 times faster signature generation engine compared to the matrix variant. At the same time verification is about 3-9 times faster. Thus, we show that the proposed constructions are quite efficient and hence competitive regarding the performance.

1.2 Related Work

The construction of lattice based signature schemes appeared to be a big challenge up to the last couples of years. This is due to the absence of practical constructions enjoying provable security. First constructions, however, such as GGH [GGH97] and NTRU Sign [HHGP$^+$03] were completely broken. This fundamentally changed in 2008 by introducing the GPV signature scheme by Gentry et al. [GPV08] and the one time signature LM-OTS by Micciancio and Lyubashevsky [LM08]. The latter one operates in ideal lattices which allows for faster computations and smaller key sizes while providing provable security. When using Merkle Trees one can transform LM-OTS into a full signature scheme. The subsequent works [Lyu08,Lyu09] build upon the one time signature scheme using the Fiat-Shamir transform [FS87]. Recently, Lyubashevsky proposed an efficient construction [Lyu12] that performs very well on hardware [GLP12].
The hash-and-sign approach in turn was reconsidered in [GPV08] leading to constructions that admit security based on the hardness of the SIS Problem. Specifically, they aim at building a uniform random matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}$ endowed with a trapdoor $\mathbf{S} \in \mathbb{Z}^{m \times m}$ in such a way that $\mathbf{S}$ has small entries and $\mathbf{A} \cdot \mathbf{S} \equiv 0 \mod q$ holds. By means of the secret matrix $\mathbf{S}$ a signer can produce short preimages $\mathbf{x}$ for the hash value $H(\mu)$ of a message μ to be signed such that $\mathbf{Ax} \equiv H(\mu)$. The quality of the secret matrix immediately transfers to the quality of the signatures and hence plays a major role for assessing the security.

Therefore, improving the algorithms for key generation is an ongoing research objective. Such constructions were considered for the first time in [Ajt99] and later on improved by [AP09,Pei10], but unfortunately they are inefficient and thus not suitable for practice. This is because the involved algorithms are complex and expensive in terms of space and runtime. However, Micciancio and Peikert recently proposed in [MP12] an elegant trapdoor construction which allows for fast signature generation while providing an improved output quality.

1.3 Organization

This paper is structured as follows. In Section 3 we introduce the GPV signature scheme together with the most recent trapdoor construction [MP12]. Furthermore, we provide a ring variant for this construction in Section 3.4. Section 4 contains a detailed description of our implementation and optimizations. In Section 5 we present the experimental results and their analysis.

2 Preliminaries

2.1 Notation

We will use the polynomial rings $R = \mathbb{Z}[x]/\langle f(x)\rangle$ and $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$ such that $f(x)$ is a monic and irreducible polynomial over $\mathbb{Z}$ and q denotes the modulus. Throughout this paper we will mainly consider the case $q = 2^k,\ k > \mathbb{N}$. For the ring-LWE problem we consider the cyclotomic polynomials, such as $f(x) = x^n + 1$ for n being a power of 2. The m-th cyclotomic polynomial with integer coefficients is the polynomial of degree $n = \phi(m)$ whose roots are the primitive m-th roots of unity.

We denote ring elements by boldface lower case letters e.g. $\mathbf{p}$, whereas for vectors of ring elements we use $\hat{\mathbf{p}}$. For a vector $\mathbf{v} \in \mathbb{R}^n$, a positive real s, and a lattice $\Lambda \subset \mathbb{R}^n$, let $D_{\Lambda,\mathbf{v},s}$ denote the n-dimensional discrete Gaussian distribution over Λ, centered at $\mathbf{v}$, with parameter s. For $x \in \Lambda$, the distribution $D_{\Lambda,\mathbf{v},s}$ assigns the probability $D_{\Lambda,\mathbf{v},s}(\mathbf{x}) := \rho_{\mathbf{v},s}(\mathbf{x}) / \sum_{\mathbf{z}\in\Lambda} \rho_{\mathbf{v},s}(\mathbf{z})$ with $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp\left(-\pi \left\|\mathbf{x}-\mathbf{v}\right\|^2 / s^2\right)$. For brevity we write $D_{\Lambda,s}$ for $D_{\Lambda,\mathbf{0},s}$ and ρ_s for $\rho_{\mathbf{0},s}$. Micciancio and Regev introduced the smoothing parameter in [MR04]:

Definition 1. *For any n-dimensional lattice Λ and positive real $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest real $s > 0$ such that $\rho_{1/s}(\Lambda^*\backslash\{0\}) \leq \epsilon$*

The matrix $\tilde{\mathbf{B}}$ stands for the Gram-Schmidt orthogonalized basis of the basis matrix $\mathbf{B}$. And $\|\mathbf{B}\|$ denotes the matrix norm of matrix $\mathbf{B}$. By $[\mathbf{A}|\mathbf{B}]$ we define the matrix obtained by the concatenation of the matrices $\mathbf{A}$ and $\mathbf{B}$.

3 Trapdoor Signatures

The signature scheme due to Gentry, Peikert and Vaikuntanathan [GPV08] consists mainly of sampling a preimage from a hash function featured with a trapdoor. The security of this construction is based on the hardness of ℓ_2-SIS. In [MP12] Micciancio and Peikert provide a new trapdoor notion that improved all relevant bounds of the previous proposals [Ajt99,AP09].

3.1 Description of the matrix version [GPV08,MP12]

Similar to the constructions of [Ajt99,AP09], the authors of [MP12] start with a uniform random matrix $\bar{\mathbf{A}} \in \mathbb{Z}^{n\times\bar{m}}$ and extend it to a matrix $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{G}-\bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}^{n\times m}$ via deterministic transformations. The main idea behind this proposal is to use a primitive matrix $\mathbf{G} \in \mathbb{Z}^{n\times\omega}$, which has the property of generating $\mathbb{Z}_q^n$ and for which one can easily sample preimages. Due to the nice structure of this matrix one can find a basis $\mathbf{S} \in \mathbb{Z}^{\omega\times\omega}$ satisfying the congruence relation $\mathbf{G}\cdot\mathbf{S} \equiv \mathbf{0} \mod q$.

Below we provide the main algorithms of the GPV signature scheme in conjuction with the trapdoor construction from [MP12]:

KeyGen$(1^n) \to (\mathbf{A},\mathbf{R})$: Sample $\bar{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_q^{n\times\bar{m}}$ and $\mathbf{R} \xleftarrow{\$} D$ such that $\mathbf{R} \in \mathbb{Z}^{\bar{m}\times\lceil\log_2(q)\rceil\cdot n}$ and D (typically $D_{\mathbb{Z}^{\bar{m}\times\lceil\log_2(q)\rceil\cdot n},\alpha q}$) is a distribution which depends on the instantiation [MP12]. Output the signing key $\mathbf{R}$ and the verification key $\mathbf{A} = [\bar{\mathbf{A}}|\mathbf{G}-\bar{\mathbf{A}}\mathbf{R}] \in \mathbb{Z}_q^{n\times m}$ where $\mathbf{G}$ is a primitive matrix.

Sign$(\mu,\mathbf{R}) \to \mathbf{x} \in \mathbb{Z}^m$: Compute the syndrome $\mathbf{u} = H(\mu)$, sample $\mathbf{p} \leftarrow D_{\mathbb{Z}^m,\sqrt{\boldsymbol{\Sigma}_\mathbf{p}}}$ and determine the perturbed syndrome $\mathbf{v} = \mathbf{u} - \mathbf{A}\cdot\mathbf{p}$. Then sample $\mathbf{z} \leftarrow D_{\Lambda_\mathbf{v}^\perp(\mathbf{G}),r}$ with $r \geq 2\cdot\sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$. Compute $\mathbf{x} = \mathbf{p} + \begin{bmatrix}\mathbf{R}\\\mathbf{I}\end{bmatrix}\mathbf{z}$ and output the signature $\mathbf{x}$.

Verify$(\mu,\mathbf{x},(H,\mathbf{A})) \to \{0,1\}$: Check whether $\mathbf{A}\cdot\mathbf{x} \equiv H(\mu)$ and $\|\mathbf{x}\|_2 \leq s\sqrt{m}$. If so, output 1 (accept), otherwise 0 (reject).

Throughout this paper fix the modulus q to be 2^k for some $k \in \mathbb{N}$ and use the primitive matrix $\mathbf{G}$ as defined in [MP12]. To this end, we start defining the primitive vector $\mathbf{g}^T := (1,2,4,\ldots,2^{k-1}) \in \mathbb{Z}_q^k$ since $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^T$. Due to its simple structure one can find an associated basis $\mathbf{S}_k$ for the lattice $\Lambda_q^\perp(\mathbf{g}^T)$ which has the following shape

$$\mathbf{S_k} = \begin{bmatrix} 2 & & & 0 \\ -1 & 2 & & \\ & \ddots & \ddots & \\ 0 & & -1 & 2 \end{bmatrix} \in \mathbb{Z}_q^{k\times k}.$$

By means of the vector $\mathbf{g}^T$ and the associated basis $\mathbf{S_k}$ one can easily create $\mathbf{S} \in \mathbb{Z}_q^{nk\times nk}$ and the primitive matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times nk}$, respectively:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}^T & & 0 \\ & \ddots & \\ 0 & & \mathbf{g}^T \end{bmatrix}, \qquad \mathbf{S} = \begin{bmatrix} \mathbf{S_k} & & 0 \\ & \ddots & \\ 0 & & \mathbf{S_k} \end{bmatrix}.$$

An optimal bound for the smoothing parameter of the lattice $\Lambda_q^\perp(\mathbf{g}^T)$ is easily obtained using the orthogonalized basis $\tilde{\mathbf{S}}_\mathbf{k} = 2 \cdot \mathbf{I}_k$. Since $\|\tilde{\mathbf{S}}\| = \|\tilde{\mathbf{S}}_\mathbf{k}\| = 2$, we have $\eta_\epsilon(\Lambda_q^\perp(\mathbf{G})) \leq r = 2 \cdot \sqrt{\ln\left(2n\left(1+\frac{1}{\epsilon}\right)\right)/\pi}$ according to [GPV08, Theorem 3.1]. Using this parameter we can bound the preimage length. Due to [Ban93, Lemma 1.5] the probability of the preimage to be greater or equal to $r \cdot \sqrt{n \cdot k}$ is bounded by $2^{-n \cdot k} \cdot \frac{1+\epsilon}{1-\epsilon}$.

Sampling Algorithms for Preimages and Perturbations In what follows we describe the preimage sampling algorithm for a syndrome $t \in \mathbb{Z}_q$ from the coset $\Lambda_t^\perp(\mathbf{g}^\top) = \{\mathbf{x} \mid \mathbf{g}^\top \cdot \mathbf{x} \equiv t \mod q\}$ using the randomized nearest plane algorithm [MP12]. Due to the nice properties of the orthogonalized basis, the algorithm reduces to a few steps with $a_0 = t$:

```
for $i = 0, \ldots, k-1$ do:
    1. $v_i \leftarrow D_{2\mathbb{Z}+a_i,r}$
    2. $a_{i+1} = (a_i - v_i)/2$
Output: $\mathbf{v} = (v_0, \ldots, v_{k-1})^T$
```

The resulting vector is $\mathbf{v} \in \Lambda_t^\perp(\mathbf{g}^\top)$ distributed as $D_{\Lambda_t^\perp(\mathbf{g}^\top),r}$. Of course, similarly one can sample preimages from $\Lambda_\mathbf{u}^\perp(\mathbf{G})$ for a syndrome vector $\mathbf{u} \in \mathbb{Z}_q^n$ by independently running n instances of the algorithm on each component of $\mathbf{u}$.
The authors provide two different types of instantiations for the trapdoor generation algorithm, namely the statistical and computational instantiation. Regarding the GPV signature scheme we used the latter one in our implementation because the dimension of $\mathbf{A}$ is much smaller. Therefore, we will always refer to the computational instantiation in the rest of this work. Such a representation can easily be achieved by generating a uniform random matrix $\tilde{\mathbf{A}} \in \mathbb{Z}^{n \times n}$ and sampling a trapdoor $\mathbf{R} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \end{bmatrix}$ from the discrete Gaussian distribution $D_{\mathbb{Z}^{2n \times nk},\alpha q}$ where $\alpha \in \mathbb{R}_{>0}$ satisfies $\alpha q > \sqrt{n}$. The resulting matrix $[\bar{\mathbf{A}}|\mathbf{G} - (\tilde{\mathbf{A}}\mathbf{R_2} + \mathbf{R_1})]$ with $\bar{\mathbf{A}} = [\mathbf{I}_n|\tilde{\mathbf{A}}]$ is an instance of decision-$LWE_{n,\alpha,q}$ and hence pseudorandom when ignoring the identity submatrix.
Applying the framework of [GPV08] requires to sample a spherically distributed preimage for a given syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ using Gaussian sampling algorithms and the trapdoor $\mathbf{R}$. In fact, the spherical distribution is a common tool to make the distribution of the signature independent from the secret key. The Gaussian sampling algorithm mainly consists of two parts. The first part involves the trapdoor $\mathbf{R}$ which is used to transform a sample $\mathbf{x}$ from the set $\Lambda_\mathbf{u}^\perp(\mathbf{G})$ with parameter $r \geq \|\tilde{\mathbf{S}}\| \cdot \sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$ to a sample $\mathbf{y} = \begin{bmatrix} \mathbf{R} \\ \mathbf{I} \end{bmatrix} \cdot \mathbf{x}$ of the set $\Lambda_\mathbf{u}^\perp(\mathbf{A})$.

Due to the fact that $\begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}$ is not a square matrix and the non-spherical covariance $\mathbf{COV} = r^2 \begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} [\mathbf{R}^T\ \mathbf{I}]$ is not of full-rank, the distribution of $\mathbf{y}$ is skewed and hence leaks information about the trapdoor. An attacker could collect samples and reconstruct the covariance matrix. Therefore, we need the second part to correct this flaw. This can be done by adding perturbations from a properly chosen distribution. Using the convolution technique from [Pei10], we can choose a parameter s in such a way that s^2 is slightly larger than the largest absolute eigenvalue of the covariance $\mathbf{COV}$ and generate Gaussian pertubations $\mathbf{p} \in \mathbb{Z}^m$ having covariance $\mathbf{\Sigma_p} = s^2\mathbf{I} - \mathbf{COV}$. In order to obtain a vector $\mathbf{b}$ that is from a spherical Gaussian distribution with parameter s, one samples a preimage $\mathbf{y}$ for an adjusted syndrome $\mathbf{v} = \mathbf{u} - \mathbf{Ap}$ from $\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A})$. The vector $\mathbf{b} = \mathbf{p} + \mathbf{y}$ provides a spherical distributed sample satisfying $\mathbf{Ab} \equiv \mathbf{u} \mod q$.

Parameters When applying the framework of [RS10] we get Table 1 which contains different parameter sets with their corresponding estimated sub-lattice attack dimension d and the more relevant estimated Hermite factor δ. The SIS norm bound is denoted by ν. Columns marked with $\star$ provide according to [GPV08, Proposition 5.7] additional worst-case to average-case hardness satisfying $q \geq \nu \cdot \omega(\sqrt{n \log_2(n)})$. The parameters of the scheme should be set in such a way that $\delta \approx 1.0064$ in order to ensure about 100 bits of security [RS10]. In Table 4 (see Appendix A.1) we provide a guideline of how to select parameters in the matrix and ring variant (Construction 1).

n	128	128*	256	256*	284	284*	384	384*	484	484*	512	512*	1024	1024
k	24	27	24	27	24	28	24	29	24	29	24	30	27	30
m	3328	3712	6656	7424	7384	8520	9984	11136	12584	15004	13312	16384	29696	32768
q	2^{24}	2^{27}	2^{24}	2^{27}	2^{24}	2^{29}	2^{24}	2^{29}	2^{24}	2^{29}	2^{24}	2^{30}	2^{27}	2^{30}
d	324	346	586	659	650	758	838	1013	1057	1221	1118	1336	2305	2561
ν	4.8e5	5.4e5	1.3e6	1.5e6	1.6e6	1.8e6	2.5e6	3.0e6	3.5e6	4.3e6	3.9e6	4.8e6	1.2e7	1.3e7
δ	1.0203	1.0183	1.0117	1.0106	1.0108	1.0095	1.0085	1.0072	1.0070	1.0060	1.0067	1.0055	1.0034	1.0031
λ bits	< 75	< 75	75	78	78	82	86	94	95	103	97	108	148	158

Table 1. Parameter sets with the corresponding estimated sublattice attack dimensions d and Hermite factors δ according to [RS10].

Different to [MR08] the approach taken in [RS10] requires to determine the optimal sub-dimension $d = \{x \in \mathbb{Z} \mid q^{2n/x} \leq \nu\}$ of the matrix $\mathbf{A}$ consisting of m columns and n rows. The lattice $\Lambda_q^{\perp}(\mathbf{A}')$ generated by $\mathbf{A}'$ when leaving out $m - d$ columns from $\mathbf{A}$ has still determinant q^n with very high probability. This means that a solution $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A}')$ with $\|\mathbf{v}\| \leq \nu$ can easily be tranformed to the vector $(\mathbf{v}, \mathbf{0})$ such that $\mathbf{A} \cdot (\mathbf{v}, \mathbf{0}) \equiv 0 \mod q$ holds. For a given d we obtain the Hermite factor $\delta = 2^{n \cdot \log_2(q)/d^2}$ implying that a sufficiently good HSVP solver can find vectors $\mathbf{v} \in \Lambda_q^{\perp}(\mathbf{A}')$ bounded by $q^{2n/d}$. From the Hermite factor one can compute the effort $T(\delta)$ required to solve $\delta - \mathsf{HSVP}$ according to [RS10, Conjecture 3]. Subsequently, one maps the result to the corresponding security levels (e.g. see [RS10, Table 2]).

3.2 The Ring Setting

In [MP12] the authors state that the construction can be adapted to the ring setting in such a way that the elements of the primitive vector $\mathbf{g}^\top$ are considered as ring elements of $R_q = \mathbb{Z}_q[X]/\phi_m(X)$ rather than $\mathbb{Z}_q$, where $\phi_m(X)$ is the m-th cyclotomic polynomial. In the following section we present our construction of this idea and show that a polynomial matrix $\hat{\mathbf{G}}$ as in the matrix case is indeed not needed. This results in a more efficient instantiation.

Construction 1 The public key is generated by drawing k samples $(\bar{\mathbf{a}}_i, \bar{\mathbf{a}}_i\mathbf{r}_i + \mathbf{e}_i)$ from the ring-LWE distribution. By this, we obtain a public key that is pseudorandom and enjoys the hardness of ring-LWE. Following [ACPS09] one can use the error distribution in order to sample the trapdoor polynomials $\hat{\mathbf{r}} \in R_q^k$ and $\hat{\mathbf{e}} \in R_q^k$. This does not incur any security flaws. Indeed, this property is essential for the signature scheme to work due to the need for smaller secret keys. As in the matrix variant one can use only one uniformly distributed sample $\bar{\mathbf{a}}_1$ rather than a set in $\mathbf{A}$. By a standard hybrid argument the hardness of distinguishing $\bar{\mathbf{a}}_1\mathbf{r_i} + \mathbf{e_i}$ from uniformly distributed samples can be reduced to decision ring-LWE [BPR12]. Thus, we obtain a public key of the following shape:

$$\mathbf{A} = [\mathbf{1},\ \bar{\mathbf{a}}_1,\ \mathbf{g}_1 - (\bar{\mathbf{a}}_1\mathbf{r}_1 + \mathbf{e}_1),\ \dots\ ,\ \mathbf{g}_k - (\bar{\mathbf{a}}_1\mathbf{r}_k + \mathbf{e}_k)]$$

Similar to the matrix version $\hat{\mathbf{g}}^\top = [1, \cdots, 2^{k-1}]$ defines the primitive vector of polynomials where each component is considered as a constant polynomial. Sampling from $\Lambda_{\mathbf{u}}^\perp(\hat{\mathbf{g}}^\top) = \{\hat{\mathbf{x}} \in R_q^k \mid \mathbf{g}_1\mathbf{x}_1 + \cdots + \mathbf{g}_k\mathbf{x}_k = \mathbf{u}\ \}$ is performed as in the matrix case with $\mathbf{y}^\top = [\mathbf{x}_1^{(0)},\ \dots\ , \mathbf{x}_k^{(0)},\ \dots\ , \mathbf{x}_1^{(n)},\ \dots\ , \mathbf{x}_k^{(n)}]$ satisfying $\mathbf{Gy} \equiv \mathbf{u} \mod q$, where $\mathbf{x}_j^{(i)}$ is the i-th coefficient of the j-th polynomial. The resulting vector $\mathbf{y}$ is from a spherical Gaussian distribution having covariance matrix $r^2\mathbf{I}$. Sampling a preimage for a syndrome $\mathbf{u} \in R_q$ requires to sample polynomials $\hat{\mathbf{x}} = (\mathbf{x}_1, \dots, \mathbf{x}_k)$ from $\Lambda_{\mathbf{u}}^\perp(\hat{\mathbf{g}}^\top)$. These are then used to construct the preimage $\hat{\mathbf{z}} = [\sum_{i=1}^{k} \mathbf{e}_i\mathbf{x}_i,\ \sum_{i=1}^{k} \mathbf{r}_i\mathbf{x}_i,\ \mathbf{x}_1,\ \dots\ ,\ \mathbf{x}_k] \in R_q^{k+2}$. It can easily be verified, that $\mathbf{A}\hat{\mathbf{z}} \equiv \mathbf{u}$ holds. With the same arguments as in the matrix case we need to add some perturbation to transform the skewed distribution into a spherical one. Since we mainly operate on rings modulo $x^n + 1$ with n a power of two, multiplication of polynomials $\mathbf{r}_i\mathbf{x}_i$ corresponds to matrix multiplication $\mathbf{Rot}(\mathbf{r}_i)\mathbf{x}_i$. The matrix $\mathbf{Rot}(\mathbf{r}_i)$ consists of n columns $[\mathbf{r}_i, \mathrm{rot}(\mathbf{r}_i), \cdots, \mathrm{rot}^{n-1}(\mathbf{r}_i)]$ with $\mathrm{rot}(\mathbf{y}) = [-y_{n-1}, y_0,\ \dots\ , y_{n-2}]$ defining the rotation in anti-cyclic integer lattices. Of course, other irreducible polynomials are also possible, but have the drawback of larger expansion factors which imply increased preimage lengths. The covariance matrix of the preimage has the following shape:

$$\mathbf{COV} = r^2 \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R_1^\top} & \mathbf{R_2^\top} & \mathbf{I} \end{bmatrix}$$

with $\mathbf{R_1} = [\mathbf{Rot}(\mathbf{e}_1) \mid \ldots \mid \mathbf{Rot}(\mathbf{e}_k)]$ and $\mathbf{R_2} = [\mathbf{Rot}(\mathbf{r}_1) \mid \ldots \mid \mathbf{Rot}(\mathbf{r}_k)]$ respectively. One observes, that the computation of this matrix is very simple since matrix multiplication corresponds to polynomial multiplication with $\beta(\mathbf{x}) = [x_1, -x_n, -x_{n-1}, \ldots, -x_2]$ which is the first row of $\mathbf{Rot}(\mathbf{x})$:

$$\mathbf{COV} = r^2 \begin{bmatrix} \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{e}_i \beta(\mathbf{e}_i)) & \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{e}_i \beta(r_i)) & \mathbf{R_1} \\ \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{r}_i \beta(\mathbf{e}_i)) & \mathbf{Rot}(\sum_{i=1}^{k} \mathbf{r}_i \beta(\mathbf{r}_i)) & \mathbf{R_2} \\ \mathbf{R_1}^\top & \mathbf{R_2}^\top & \mathbf{I} \end{bmatrix} .$$

Now one can use the techniques from the previous section in order to generate perturbations. A perturbation vector $\mathbf{p} \in \mathbb{Z}^{n(k+2)}$ is then split into $k+2$ parts of length n. Each part corresponds to a perturbation polynomial $\mathbf{p}_i \in R_q$. In order to provide a preimage for a syndrome polynomial $\mathbf{u}$ one samples perturbations $\mathbf{p}_1, \ldots, \mathbf{p}_{k+2} \in R_q$ as shown before. Then we create sample polynomials $\hat{\mathbf{x}}$ from $\Lambda^{\perp}_{\mathbf{u}-\mathbf{Ap}}(\hat{\mathbf{g}}^\top)$. The resulting preimage $\hat{\mathbf{z}}$ is then spherically distributed:

$$\hat{\mathbf{z}} = \left[\mathbf{p}_1 + \hat{\mathbf{e}} \cdot \hat{\mathbf{x}},\, \mathbf{p}_2 + \hat{\mathbf{r}} \cdot \hat{\mathbf{x}},\, \mathbf{p}_3 + \mathbf{x}_1, \ldots\ , \mathbf{p}_{k+2} + \mathbf{x}_k\right] .$$

Now we give a short description of how to instantiate the ring-LWE problem and how to sample the secret keys $\mathbf{r}_i$ and $\mathbf{e}_i$ for $1 \leq i \leq k$ according to [DD12]. The authors provide different from the work [LPR10] a relatively simple ring-LWE setting avoiding the work in the dual ring $R_q^\vee$ or the H-Space [LPR10] which turns out to be more convenient in certain applications. Following the paper of [BLP$^+$13] we can take q to be a power of two as in the matrix variant. Such choices are more suitable for practice since the nice sampling algorithms introduced in the previous section are applicable. A prime number would involve costly sampling procedures which lead to a slower signature generation engine. As stated in [ACPS09] it is possible to generate both the secret key $\mathbf{r}_i$ and $\mathbf{e}_i$ from the same error distribution without affecting the security. Indeed, this property is important in order to make the trapdoor construction work based on the ring-LWE assumption. Specifically, we need small keys to provide short preimages. If one operates in the ring $\mathbb{Z}_q[X]/(X^n + 1)$ with n a power of two, the coefficients of both $\mathbf{r}_i$ and $\mathbf{e}_i$ are chosen from the Gaussian distribution on the rationals and then rounded to the nearest integers. In particular, the polynomials $\mathbf{r}_i$ and $\mathbf{e}_i$ are distributed as $\lceil D_{\mathbb{Q}^n,c} \rfloor$ for $c = \sqrt{n}\alpha q(\frac{nl}{2\log(nl/2)})^{1/4}$ where l is the number of samples and $\alpha q > \omega(\sqrt{\log 2n})$. In practical applications one can omit the last term [DD12] or set $l = 1$ due to the fact that a possible adversary can always create own samples by using $\bar{\mathbf{a}}_\mathbf{1}$. For other choice of cyclotomic polynomials Φ_m it is possible to sample the trapdoor polynomials in extension rings according to [DD12, Theorem 2]. But a better approach is to use the framework presented in [LPR13] since it allows to work in arbitrary cyclotomic rings without incurring any ring-dependent expansion factor. We will provide such a construction in the full version of this paper.

Construction 2 We briefly explain another ring construction that is derived from [Mic07]. Take $k = \lceil \log_2 q \rceil$ and $\bar{m} = O(\log_2(n))$. Then select $\bar{m}$ uniformly random polynomials $\hat{\mathbf{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_{\bar{m}}] \in R_q^{\bar{m}}$. Define by $h_{\hat{\mathbf{a}}}(\hat{\mathbf{x}}) = \sum_{i=1}^{\bar{m}} \mathbf{a}_i \mathbf{x}_i$ a generalized compact knapsack. Furthermore choose k vectors $\hat{\mathbf{r}}_i$ for $1 \leq i \leq k$, each consisting of $\bar{m}$ random polynomials $\mathbf{r}_{i1}, \ \ldots \ , \ \mathbf{r}_{i\bar{m}}$ of degree $n-1$ with small coefficients. By [SSTX09, Lemma 6], which is an adapted variant of the regularity lemma of [Mic07], the function values $\mathbf{a}_{\bar{m}+i} = h_a(\hat{\mathbf{r}}_i)$ with $1 \leq i \leq k$ are essentially uniformly distributed. Thus, we can create an almost uniformly random vector of polynomials $\mathbf{A}$ endowed with the trapdoor $\hat{\mathbf{r}}_i \in R_q^{\bar{m}}$ where $1 \leq i \leq k$:

$$\mathbf{A} = [\mathbf{a}_1, \ \ldots \ , \ \mathbf{a}_{\bar{m}}, \ \mathbf{g}_1 - \mathbf{a}_{\bar{m}+1}, \ \ldots \ , \ \mathbf{g}_k - \mathbf{a}_{\bar{m}+k}] \,.$$

To generate a preimage of a given syndrome polynomial $\mathbf{u} \in R_q$, one has to sample a vector $\hat{\mathbf{x}} \in \Lambda_{\mathbf{u}}^{\perp}(\hat{\mathbf{g}}^{\top})$ using the methods from above. As one can easily verify, the vector $\hat{\mathbf{y}} = [\hat{\mathbf{r}}_1 \mathbf{x}_1, \ \ldots \ , \ \hat{\mathbf{r}}_k \mathbf{x}_k \ , \ \mathbf{x}_1, \ \ldots \ , \mathbf{x}_k]$ is a preimage of the syndrome $\mathbf{u}$ for $\mathbf{A}$. Using the techniques from the descriptions before, one can produce spherically distributed samples.

4 Improvements and Implementation Details

In our implementation we have to face several challenges that affect the performance of the signature scheme both in the matrix and ring variant. In the following sections we give a detailed description of our improvements and implementation results.

4.1 Computation of the Covariance matrix

Firstly, we observed that the computation of the covariance matrix **COV** is too expensive in terms of running time. Since the basis matrix **COV** is sparse, we were able to significantly reduce the computational efforts. It can be split into four parts as below. The only block to be computed is the symmetric matrix $\mathbf{R}\mathbf{R}^T$.

$$\mathbf{COV} = r^2 \begin{bmatrix} \mathbf{R}\mathbf{R}^T & \mathbf{R} \\ \mathbf{R}^T & \mathbf{I} \end{bmatrix}$$

In the ring variant the computation of the covariance matrix is much faster because multiplication is performed in polynomial rings as explained in the description. Running these parts in parallel offers another source of optimization.

4.2 Estimating the parameter s

As in [MP12] one sets the parameter s large enough such that it is independent from a specific trapdoor. In particular, s is chosen to be not smaller than $\sqrt{s_1(\mathbf{R})^2 + 1} \cdot \sqrt{6} \cdot a$, where $s_1(\mathbf{R})$ denotes the largest singular value of the secret key $\mathbf{R}$ and a is selected as above. The perturbation covariance matrix

$\mathbf{\Sigma_p} = s^2\mathbf{I}_m - \mathbf{COV}$ is well-defined, if one selects s such that $s > s_1(\begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix}) \cdot r$ is satisfied. Since $\mathbf{R}$ is a subgaussian random variable, the matrix $\mathbf{R}$ satisfies $s_1(\mathbf{R}) \leq C \cdot (\sqrt{2n} + \sqrt{n \cdot k} + 4.7) \cdot \alpha q$ except with probability $\approx 2^{-100}$ according to [MP12, Lemma 2.9]. The universal constant C is very close to $1/\sqrt{2\pi}$.

4.3 Generation of Perturbation Vectors

One of the main ingredients of the signature scheme is the idea of creating perturbations [MP12] in order to get spherically distributed preimages that do not carry any information about the secret key. A perturbation vector is generated by means of the distribution $D_{\mathbb{Z}^m,\sqrt{\mathbf{\Sigma_p}}}$ which outputs random vectors from $\mathbb{Z}^m$ with covariance matrix $\mathbf{\Sigma_p}$. By [Pei10] this can be achieved by sampling a vector $\mathbf{p}$ according to $\lceil \sqrt{\mathbf{\Sigma_p} - a^2\mathbf{I}} \cdot D_1^m \rfloor_a$, where D_1^m denotes the m-dimensional Gaussian distribution. Each vector sampled from D_1^m has entries coming from the standard continuous Gaussian distribution with parameter 1. $\lceil \cdot \rfloor_a$ denotes the randomized rounding operation from [Pei10] with parameter $a = r/2 \geq \sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$, which rounds each coordinate of the vector independently to a nearby integer using the discrete Gaussian distribution. The generation of perturbation vectors requires the square root computation $\sqrt{\mathbf{\Sigma_p} - a^2\mathbf{I}}$. Below we discuss one method for this purpose and provide improvements through a better analysis.

4.4 Square Root Computation

The Cholesky decomposition splits any positive definite matrix $\mathbf{M}$ into the product of a lower triangular matrix and its conjugate transpose, i.e. $\mathbf{M} = \mathbf{L} \cdot \mathbf{L}^T$, and runs in time $O(m^3) = O((k+2)^3 n^3)$. If one selects $k = 19$, then the constant factor grows by 9261, which is very high compared to $n = 256$. The Cholesky decomposition is needed to generate perturbations that have covariance matrix $\mathbf{\Sigma_p}$, where $\sqrt{\mathbf{\Sigma_p}}$ is the Cholesky matrix. An algorithm for the Cholesky decomposition is shown in the Appendix A.2 (Algorithm 1). When decomposing the matrix $\mathbf{\Sigma_p} - a^2\mathbf{I}$ into its roots, one can improve the running time by our modified Cholesky decomposition taking into account the $n^2k^2 - n \cdot k$ zero entries, meaning that one can skip line 8 in Algorithm 1 whenever l_{ik} or l_{jk} is known to be zero. Due to the sparsity of $\mathbf{\Sigma_p} - a^2\mathbf{I}$ this occurs very often. We call this optimized algorithm variant 1.
Although this optimization in variant 1 noticeably improves the timings of key generation, the algorithm is still inefficient and is the main source of slow key generation. Moreover, the resulting perturbation matrix is dense and has no structure, which leads to high memory claims in order to store the matrix of floating entries and to worse signature generation running times. This is due to the fact that each generation of a perturbation vector requires to multiply a huge triangular matrix consisting of multi-precision floating point entries with a floating point vector. To circumvent this problem we applied a pivoting strategy followed by the Block Cholesky decomposition, meaning that we permute the covariance matrix such that $\mathbf{P}\mathbf{\Sigma_p}\mathbf{P}^\top = \mathbf{\Sigma}'_\mathbf{p}$.

This corresponds to left multiplication of the permutation matrix $\mathbf{P} = \begin{bmatrix} 0 & \mathbf{I}_{nk} \\ \mathbf{I}_{2n} & 0 \end{bmatrix}$ to the public key $\mathbf{A}$. It is obvious that this transformation does not cause any security flaws because it is a simple reordering. The advantage of using $\mathbf{P}$ is a perturbation covariance matrix $\mathbf{\Sigma}'_{\mathbf{p}}$ with a nice structure which enables us to work with Schur complements [Zha10] in a very efficient way:

$$\mathbf{\Sigma}'_{\mathbf{p}} = s^2\mathbf{I}_m - r^2 \begin{bmatrix} \mathbf{I}_{nk} & \mathbf{R}^\top \\ \mathbf{R} & \mathbf{R}\mathbf{R}^\top \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I}_{nk} \\ \mathbf{I}_{2n} & 0 \end{bmatrix} \mathbf{\Sigma}_{\mathbf{p}} \begin{bmatrix} 0 & \mathbf{I}_{nk} \\ \mathbf{I}_{2n} & 0 \end{bmatrix}^\top .$$

Therefore we get an algorithm which outperforms the optimized cholesky decomposition applied on the non-permuted matrix by a factor of 30-190. Furthermore, we obtain a signature generation engine which yields a factor improvement of 2-6 in the ring variant. This is due to the sparse matrix and its nice structure. In both the key and signature generation steps the factor grows as n increases. In general the Schur complement is defined as follows:

Lemma 1. *Let the matrix* $\mathbf{S_i} = \left[\begin{array}{c|c} b_i & \mathbf{h}_i^\top \\ \hline \mathbf{h}_i & \mathbf{C}_i \end{array}\right] \in \mathbb{R}^{m-i \times m-i}$ *be symmetric positive definite with* $b_i > 0$*. Then the Schur complement* $\mathbf{S}_{i+1} := \mathbf{C}_i - \frac{1}{b_i}\mathbf{h}_i\mathbf{h}_i^\top \in \mathbb{R}^{m-i-1 \times m-i-1}$ *is well-defined and also symmetric positive definite.*

This decomposition is successively applied on the submatrices $\mathbf{S_i} \in \mathbb{R}^{m-i \times m-i}$. Doing this, one obtains an efficient method to construct the columns of the matrix $\sqrt{\mathbf{\Sigma}'_{\mathbf{p}} - a^2\mathbf{I}}$. The first nk colums $\frac{1}{\sqrt{b}} \cdot \begin{bmatrix} b \cdot \mathbf{I} \\ \mathbf{R} \end{bmatrix} \in \mathbb{R}^{m \times nk}$ for $b = s^2 - r^2 - a^2 = s^2 - 5a^2$ involve only a simple scaling operation. Therefore, we need no additional memory in order to store these columns. Due to the sparse columns multiplication involves only the non-zero columns $(\mathbf{R})_i$ of the matrix $\mathbf{R} = \begin{bmatrix} \mathbf{R_1} \\ \mathbf{R_2} \end{bmatrix}$. Thus, transformations are focused only on the $(2n \times 2n)$ matrix:

$$\mathbf{S}_{nk} = (s^2-a^2)\mathbf{I} - r^2\mathbf{R}\mathbf{R}^\top - \frac{1}{b}\sum_{i=1}^{nk}(\mathbf{R})_i(\mathbf{R})_i^\top = (s^2-a^2)\mathbf{I} - (r^2+\frac{1}{b})\mathbf{R}\mathbf{R}^\top \in \mathbb{R}^{2n \times 2n} .$$

The last sum of vector products reduces to the simple scaling operation $\frac{1}{b}\mathbf{R}\mathbf{R}^\top$. Thus, one can save the costly vector product computations. When continuing the decomposition on the remaining matrix $\mathbf{S}_{nk}$ one obtains the Cholesky decomposition. One can easily verify that

$$\mathbf{X}\mathbf{X}^\top = \mathbf{\Sigma}'_{\mathbf{p}} - a^2\mathbf{I}, \;\; \mathbf{X} = \begin{bmatrix} \sqrt{b}\mathbf{I_{nk}} & 0 \\ \frac{\mathbf{R}}{\sqrt{b}} & \mathbf{L} \end{bmatrix}$$

holds. Consequently one needs only to store $n(2n+1)$ floating point entries of the last part $\mathbf{L} = Decomp(\mathbf{S_{nk}})$ instead of $m(m+1)/2$ in the case without permutation. For instance, this induces an improvement factor of $m(m+1)/2n(2n+1) \approx 240$ for $n = 512$ and $k = 29$. A nice sideeffect of this transformation is a much faster algorithm for generating perturbations since the number of operations drastically decreases as the factor grows. In the matrix version, one makes use of the sparse decomposition matrix. In particular $\sqrt{\mathbf{\Sigma}_{\mathbf{p}} - a^2\mathbf{I}} \cdot D_1^m$ is reduced to the simple scaling operation of $\sqrt{b} \cdot D_1^{nk}$ and the computation $\left[\frac{1}{\sqrt{b}}\mathbf{R} \,\middle|\, \mathbf{L}\right] \cdot D_1^m$. Especially in the ring version we preserve the nice properties of polynomial multiplication and therefore use only the scaled set of trapdoor polynomials $\frac{1}{\sqrt{b}}\mathbf{e}_i$, $\frac{1}{\sqrt{b}}\mathbf{r}_i$ and the lower triangular matrix $\mathbf{L} = \begin{bmatrix}\mathbf{L_1} \\ \mathbf{L_2}\end{bmatrix}$ in order to generate perturbations. Specifically, one obtains the perturbation vector $\mathbf{p} = [\mathbf{p_1}|\mathbf{p_2}|\mathbf{p_3}] \in \mathbb{Q}^{(k+2)n}$ with $\mathbf{p_1} = \sqrt{b} \cdot D_1^{nk}$, $\mathbf{p_2} = \frac{1}{\sqrt{b}} \sum_{i=1}^{k} \mathbf{r}_i D_1^n + \mathbf{L_1} \cdot D_1^n$ and $\mathbf{p_3} = \frac{1}{\sqrt{b}} \sum_{i=1}^{k} \mathbf{r}_i D_1^n + \mathbf{L_2} \cdot D_1^n$. Thus, we get a fast signature generation algorithm which is about three times faster than its matrix analogue. It is also worth to mention that these operations can also be executed in parallel.

4.5 Sampling

For sampling discrete Gaussian distributed integers in the key generation step we used the inversion transform method rather than rejection sampling because the number of stored entries is small and can be deleted afterwards. This improves the running times of the sampling step significantly. In particular, suppose the underlying parameter is denoted by s. We precompute a table of cumulative probabilties p_t from the discrete Gaussian distribution with $t \in \mathbb{Z}$ in the range $[-\omega(\sqrt{\log n}) \cdot s, \omega(\sqrt{\log n}) \cdot s]$. We then choose a uniformly random $x \in [0, 1)$ and find t such that $x \in [p_{t-1}, p_t]$. This can be done using binary search. The same method is applied when sampling preimages from the set $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{G})$ with parameter r. This parameter is always fixed and relatively small. Storing this table takes about 150 Bytes of memory. In this case signature generation is much faster than with simple rejection sampling. But, unfortunately, this does not apply in the randomized rounding step because the center always changes and thus involves a costly recomputation of tables after each sample. Therefore we used rejection sampling from [GPV08] instead. As for sampling continuous Gaussians with parameter $t = 1$, we used the Ziggurat algorithm [MT84] which is one of the fastest algorithms to produce continuous Gaussians. It belongs to the class of rejection sampling algorithms and uses precomputed tables. When operating with multiprecision vectors such as sampling continuous random vectors one should use at least λ bits of precision for a cryptographic scheme ensuring λ bits of security (e.g. 16 bytes floating points for $\lambda = 100$).

4.6 Random Oracle Instantiation

For the GPV signature scheme a random oracle $H(\cdot)$ is required which on an input message x outputs a uniformly random response $H(x)$ from its image space. In most practical applications this is achieved by a cryptographic hash function together with a pseudorandom generator which provides additional random strings in order to extend the output length. In our implementation we used SHA256 together with the GMSS-PRNG [BDK+07] because strings of arbitrary size are mapped to vectors from $\mathbb{Z}_q^n$. Each component of the vector has at most $\lfloor \log q \rfloor$ bits.

$$Rand \leftarrow H(Seed_{in}), Seed_{out} \leftarrow (1 + Seed_{in} + Rand) \mod 2^n. \tag{1}$$

The first $Seed_{in}$ is the input message, and the function is repeated until enough random output $Rand$ is generated.

We implemented the GPV signature scheme, the trapdoor generation and sampling algorithms in C using the Fast Library for Number Theory (FLINT 2.3) and the GNU Scientific Library (GSL 1.15). FLINT comprises different data types for matrices and vectors operating in residue classes such as $\mathbb{Z}_q$ and $\mathbb{Z}_q[X]$ wheras the GSL library provides a huge variety of mathematical tools from linear algebra, that can be applied on different primitive data types. We also included the Automatically Tuned Linear Algebra Software Library (ATLAS) which is an empirical tuning system that creates an individual BLAS (Basic Linear Algebra Subprograms) library on the target platform on which the library is installed on. Specifically, this library provides optimized BLAS routines which have a significant impact on the running times of the used mathematical operations in the key and signature generation steps. So it is always recommended to include this library whenever one has to work with GSL. For the representation of matrices in $\mathbb{Z}_q^{n \times m}$ FLINT provides the data structure `nmod_mat_t` which comes into use in our implementation of the matrix version. Regarding the ring version, working with polynomials is performed by using the data structure `nmod_poly_t`. FLINT makes use of a highly optimised Fast Fourier Transform routine for polynomial multiplication and some integer multiplication operations.

5 Experimental Results

In this section we present our experimental results and compare the matrix version with the ring variant. Regarding the GPV signature scheme we used Construction 1 operating with a smaller number of polynomials compared to Construction 2. Hence, we obtain faster signature generation algorithms with a view to polynomial multiplication, generation of perturbations and sampling algorithms. We provide running times and file sizes of keys and signatures. The experiments were performed on a Sun XFire 4400 server with 16 Quad-Core AMD Opteron(tm) Processor 8356 CPUs running at 2.3GHz, having 64GB of memory and running 64bit Debian 6.0.6. We used only one core in our experiments. In most works private keys and signature sizes are estimated based on the underlying distributions ignoring the norm bound of the sampled vectors and thus lead to overestimations of signature sizes. By Lemma 2 we show that we can ignore the underlying distributions and focus solely on the norm bound. This allows us to give tighter bounds compared to previous proposals. For instance, in [Lyu12] signatures $\mathbf{y} \in \mathbb{Z}^m$ are distributed as discrete Gaussians with standard deviation σ. The estimated signature size is $m \cdot \lceil \log_2(12 \cdot \sigma) \rceil$ bits (ignoring the norm bound). In our case signatures are distributed as discrete

Gaussians with parameter s such that $\|\mathbf{y}\|_2 < s \cdot \sqrt{m}$. Using Lemma 2 the bit size needed to represent $\mathbf{y}$ is bounded by $m \cdot (1 + \lceil \log_2(s) \rceil)$ bits. The private key $\mathbf{R} \in \mathbb{Z}^{2n \times n \cdot k}$ from Section 3.1 can be viewed as a vector $\mathbf{r}$ with $2n^2k$ entries such that $\|\mathbf{r}\|_2 < \alpha q \cdot \sqrt{2n^2k}$ by [Ban93, Lemma 1.5].

Lemma 2. *Let $\mathbf{v} \in \mathbb{Z}^n$ be a vector with $\|\mathbf{v}\|_2 < b \cdot \sqrt{n}$. Then, the maximum number of bits required to store this vector is bounded by $n \cdot (1 + \lceil \log_2(b) \rceil)$.*

The proof of Lemma 2 is in the Appendix (see A.3). Below we provide two tables comparing the ring variant with the matrix variant. They contain the filesizes of the private key, public key, perturbation matrix and the signature (Table 2 bottom) as well as the running times of key generation, signature generation and verification (Table 2 top). The last line of the table reflects the improvement induced by the modification of the public key $\mathbf{A}$ and hence the covariance matrix. The improvement factor is related to the optimized Cholesky decomposition (variant 1) which makes use of the sparsity of $\mathbf{\Sigma_p}$. Indeed, the improvement factor is much higher when comparing to the original Cholesky decomposition. The impact of the discrete Gaussian samplers and the ATLAS library used in our implementation are notably but not addressed in this work.

Running times [ms]										
		Keygen			Signing			Verification		
n	k	Ring	Mat	M/R	Ring	Mat	M/R	Ring	Mat	M/R
128	24	277	984	3.6	5	9	1.8	0.6	1.4	2.3
128	27	317	1,108	3.5	6	11	1.8	0.7	1.7	2.4
256	24	1,070	5,148	4.3	12	30	2.5	1.5	5	3.3
256	27	1,144	5,728	4.1	14	36	2.5	1.7	6	3.5
512	24	4,562	28,449	5.0	27	103	3.8	3	18	6
512	27	5,354	30,458	5.1	31	125	4.0	4	21	5.3
512	29	5,732	34,607	5.4	35	136	3.8	5	22	4.4
1024	27	28,074	172,570	6.0	74	478	6.4	10	97	9.7
1024	29	30,881	198,620	6.3	81	518	6.4	11	102	9.3
Improvement factor		30-190 ↑	10 -40 ↑	-	2-6 ↑	1.4 - 2 ↑	-	-	-	-

Sizes [kB]											
		Public Key			Secret Key			Pert. Matrix	Signature		
n	k	Ring	Mat	M/R	Ring	Mat	M/R	R and M	Ring	Mat	M/R
128	24	9.4	1200	128	4.4	528	163	257	5.8	5.3	0.9
128	27	11.8	1512	128	5.0	594	163	257	6.5	5.9	0.9
256	24	18.8	4800	256	9.8	2304	236	1026	12.5	11.4	0.9
256	27	23.6	6048	256	11.0	2592	236	1026	14.1	12.8	0.9
512	24	37.5	19,200	512	21.3	9984	469	4100	26.8	24.5	0.9
512	27	47.3	24,192	512	23.9	11232	470	4100	30.1	27.4	0.9
512	29	54.4	27,840	512	25.7	12064	470	4100	32.2	29.4	0.9
1024	27	94.5	96,768	1024	51.7	48384	936	16392	63.8	58.5	0.9
1024	29	108.8	111,360	1024	55.5	51968	936	16392	68.4	62.7	0.9
Improvement factor		-	-	-	-			170 - 260	-	-	-

Table 2. Experimental results for the matrix and ring variant. By ↑ we mean that the factor grows as n increases.

By the modification we obtain a key generation engine that is about 30-190 times faster in the ring variant. For $n = 512$ and $n = 1024$ signature generation is about 3 and respectively 6 times faster. It is also worth to mention that the authors of [MP12] explain the possibility of splitting the signing algorithm into an offline and online phase. The task of generating pertubations is independent from the message to be signed, hence it is possible to generate them in advance or create many samples and store them. This obviously requires to periodically create the perturbation matrix or storing it. From a practical point of view we do not consider such a breakdown in our implementations. But indeed, generating perturbations amounts after the optimizations to more than 60 percent (see Figure 1) of the running time in the ring variant and 13-30 percent in the matrix variant. In Figure 1 we present a breakdown of the signing running time into four major parts which are the most time consuming. In particular, we differentiate the generation of perturbations $\hat{\mathbf{p}}$, sampling of $\hat{\mathbf{x}}$, computation of the syndrome polynomial $\mathbf{v} = \mathbf{A}\hat{\mathbf{p}}$, polynomial multiplications $\hat{\mathbf{e}} \cdot \hat{\mathbf{x}}$ and $\hat{\mathbf{r}} \cdot \hat{\mathbf{x}}$. By our experiments for different parameter sets we obtain Figure 1 illustrating the average measurements.

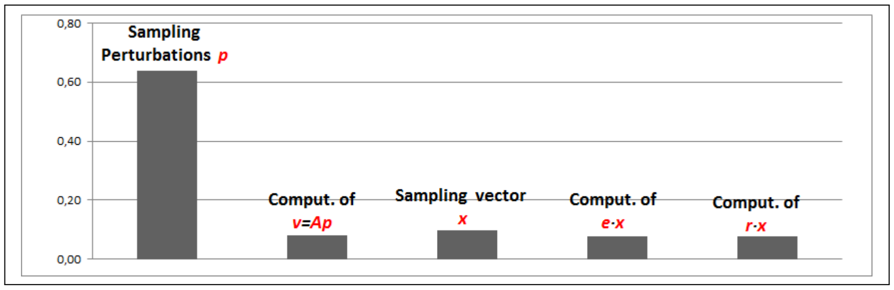

Fig. 1. Breakdown of signing running time into the major parts

In Table 3 we compare our implementation with classical signature schemes such as ECDSA, DSA and RSA for the same machine (AMD Opteron at 2.3GHz). The experiments were performed based on `openssl` implementations of the corresponding schemes. In addition, we provide implementation results of current post quantum schemes, such as the code-based signature schemes [V97,Ste94] using the Fiat-Shamir transform [FS87,ADV$^+$12]. As one observes, all classical schemes have faster signing algorithms compared to our implementation except for RSA 4096. However, our implementation has a faster signature verification engine than ECDSA and outperforms the code-based Véron und Stern signature schemes as well as the software implementations [WHCB13] of the lattice-based signature scheme [Lyu12]. Newer variants and implementations [GOPS13] are highly optimized and testify the superiority of lattice-based signature schemes over number theoretic ones.

Table 5 in the Appendix A.4 depicts the sizes of signatures, secret and public keys of the most recent lattice-based signature schemes at a glance. A look to this table reveals that the storage sizes of the GPV signature scheme are still large compared to [Lyu12,GLP12].When comparing our scheme with the ring equiva-

Scheme	Security level	Sizes [kB]			Running times [ms]	
		Public Key	Secret Key	Signature	Signing	Verification
GPV Ring (n=512, k=24)	≈ 90	37.5	21.3	26.8	27	3
GPV Ring (n=512, k=27)	≈ 100	47.3	23.9	30.1	31	4
TSS [WHCB13,Lyu12] (n=512)	80	12.8	12.9	8.1	40.7	5.6
LyuSig[GOPS13] (n=512)	100	1.5	0.3	1.2	0.3	0.02
Stern Sign.[Ste94,ADV$^+$12] (rounds=140)	80	36	0.05	25	32	23
Veron Sign.[V97,ADV$^+$12] (rounds=140)	80	36	0.05	25	31	24
RSA 2048	112	0.3	2	0.3	5.0	0.05
RSA 4096	≥ 128	0.5	4	0.5	27.5	0.14
DSA 2048	112	0.3	0.02	0.04	0.7	0.8
ECDSA 233	112	0.06	0.09	0.06	7	3.5
ECDSA 283	128	0.07	0.11	0.07	11.5	6.5

Table 3. Comparison of different signature schemes.

lent of l_2-SIS, one observes that the public key and signature sizes are about 30% higher. The secret key sizes of our implementation are even higher if one stores the perturbation matrix and does not create it for each signature (see Table 2). The optimizations due to [GLP12] furtherly improve the sizes of [Lyu12] by using more aggressive parameters. In [DDLL13] Ducas et al. present a novel signature scheme that benefits from a highly efficient bimodal discrete Gaussian sampler and a modified scheme instantiation compared to [Lyu12,GLP12]. Furthermore, they provide a scheme variant that allows key generation to be performed in a NTRU-like manner. The corresponding sizes of keys and signatures for BLISS providing 128 bits of security are also depicted in Table 5.

Acknowledgements

We would like to thank Chris Peikert and Özgür Dagdalen for the fruitful discussions as well as the reviewers of SAC.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[ADV$^+$12] Sidi Mohamed El Yousfi Alaoui, Özgür Dagdelen, Pascal Véron, David Galindo, and Pierre-Louis Cayrel. Extended security arguments for signature schemes. In *Progress in Cryptology - AFRICACRYPT 2012*, volume 7374 of *LNCS*, pages 19–34. Springer, 2012.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, LNCS, pages 1–9. Springer, 1999.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, volume 3 of *LIPIcs*, pages 75–86. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–635, 1993.

[BDK$^+$07] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *ACNS 2007*, LNCS, pages 31–45. Springer, 2007.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, 2013.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, April 2012.

[DD12] Léo Ducas and Alain Durmus. Ring-Lwe in polynomial rings. PKC'12, pages 34–51. Springer, 2012.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal gaussians. Cryptology ePrint Archive, Report 2013/383, 2013. `http://eprint.iacr.org/2013/383`.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1987.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, August 1997.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, volume 7428 of *LNCS*. Springer, 2012.

[GOPS13] Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. Software speed records for lattice-based signatures. In *PQCrypto*, 2013.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008.

[HHGP$^+$03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, JosephH. Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Topics in Cryptology CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. SPRINGER, 2003.

[Lar12] Ron Larson. *Brief Calculus: An Applied Approach*, volume 9. 2012.

[LM08] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer, March 2008.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, May 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, pages 35–54, 2013.

[Lyu08] Vadim Lyubashevsky. Towards practical lattice-based cryptography, 2008.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, December 2009.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755. Springer, April 2012.

[Mic07] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, April 2012.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381. IEEE Computer Society Press, October 2004.

[MR08] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In DanielJ. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, 2008.

[MT84] G. Marsaglia and W. Tsang. A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions. *SIAM Journal on Scientific and Statistical Computing*, 5(2):349–359, 1984.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, August 2010.

[RS10] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. 2010. `http://eprint.iacr.org/2010/137`.

[Sho97] 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.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, December 2009.

[Ste94] Jacques Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology CRYPTO 93*, volume 773 of *LNCS*, pages 13–21. Springer, 1994.

[V97] Pascal Vron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8:57–69, 1997.

[WHCB13] Patrick Weiden, Andreas Hülsing, Daniel Cabarcas, and Johannes Buchmann. Instantiating treeless signature schemes. *IACR Cryptology ePrint Archive*, 2013.

[Zha10] Fuzhen Zhang. *The Schur Complement and Its Applications*, volume 4. Springer, 2010.

A Appendix

A.1 Parameter Choices for the matrix and ring variant

	Matrix	Ring
n	e.g. $n \geq 384$ (cf. Table 1)	$n = 2^l$, $n \geq 512$
q	e.g. power of 2 with $q \geq 2^{19}$	
k	$\lceil \log_2(q) \rceil$	
m	$n(2+k)$	
c	$c = \alpha q > \sqrt{n}$	$c > \sqrt{n}\omega(\sqrt{\log(2n)})$
r	$r \geq 2 \cdot \sqrt{\ln(2n(1+\frac{1}{\epsilon}))/\pi}$	
a	$a \geq \sqrt{\ln(2(1+\frac{1}{\epsilon}))/\pi}$, e.g. a=r/2	
s	$\approx C \cdot (\sqrt{n} \cdot k + \sqrt{2n}) \cdot c \cdot r$	

A.2 Cholesky decomposition

```
Algorithm 1: Cholesky decomposition
Data: Matrix $\mathbf{L} \in \mathbb{Z}^{m \times m}$
Result: Lower triangular part of $\mathbf{L}$
for $k = 1 \rightarrow m$ do
    $l_{kk} = \sqrt{l_{kk}}$;
    for $i = k+1 \rightarrow m$ do
        $l_{ik} = l_{ik}/l_{kk}$;
    end
    for $j = k+1 \rightarrow m$ do
        for $i = j \rightarrow m$ do
            $l_{ij} = l_{ij} - l_{ik}l_{jk}$;
        end
    end
end
```

A.3 Proof of Lemma 2

Proof. We determine the maximum number of bits needed to store a vector $\mathbf{v}$ bounded by $\|\mathbf{v}\|_2 < b \cdot \sqrt{n}$ by means of Lagrange multipliers [Lar12]. The general form of Lagrange multipliers is defined by $L(v_1, \ldots, v_n) = f(v_1, \ldots, v_n) + \lambda \cdot g(v_1, \ldots, v_n)$, where $g(\cdot)$ takes into account the constraints and $f(\cdot)$ is the function to be maximized. Obviously, the maximum number of bits grows with increasing norm bound. Therefore, let $\mathbf{v} \in \mathbb{N}^n$ (ignoring the signs) be a vector such that $\|\mathbf{v}\|_2^2 = \sum_{i=1}^{n} v_i^2 = nb^2$. Now, consider the log entries of the vector $\mathbf{v}$, which are needed to determine the bit size of any vector. Applying simple logarithm rules we have $\sum_{i=1}^{n} \log_2(v_i) = \log_2(\prod_{i=1}^{n} v_i)$. Since log is monotone increasing, maximizing of log is equivalent to maximizing the product. The function giving the constraint is $g(v_1, \ldots, v_n) = nb^2 - \sum_{i=1}^{n} v_i^2$. We then maximize the function

$L(v_1,\dots,v_n,\lambda) = f(v_1,\dots,v_n) + \lambda \cdot g(v_1,\dots,v_n)$, where $f(v_1,\dots,v_n) = \prod_{i=1}^{n} v_i$.
Taking the partial derivatives we get $n+1$ equations:

$$\frac{\Delta L}{\Delta v_i} = \frac{\Delta f}{\Delta v_i} + \frac{\lambda \cdot \Delta g}{\Delta v_i} = \prod_{j=1, j\neq i}^{n} v_j - 2\lambda v_i = 0, \qquad \forall 1 \leq i \leq n$$

$$\frac{\Delta L}{\Delta \lambda} = nb^2 - \sum_{i=1}^{n} v_i^2 = 0\,.$$

By reordering the first n equations, we get $\lambda = \frac{v_1 \cdot \ldots \cdot v_{i-1} \cdot v_{i+1} \cdot \ldots \cdot v_n}{2 v_i}$, $\forall 1 \leq i \leq n$. It is easy to see that the only solution is $v_i = b$, $\forall 1 \leq i \leq n$ that satisfies all equations, because from any two out of the first n equations it follows $v_i = v_j$, $i \neq j$. By the last equation we then obtain $v_i = b$. The only extremum we obtain is $\mathbf{v} = (v_1,\dots,v_n) = (b,\dots,b)$ with $f(\mathbf{v}) = b^n$. Since we have $0 = f(\mathbf{v}') < b^n$ for the boundary points $v'_i = b \cdot \sqrt{n}$ with $v'_j = 0$ and $j \neq i$, the extremum $\mathbf{v}$ is a maximum. Therefore the maximum possible bit size required to store such a vector is bounded by $n \cdot \lceil \log_2(b) \rceil$. We need an additional bit for the sign of each entry. This concludes the proof. The proof can be extended to any p-norm $1 \leq p < \infty$. $\square$

A.4 Sizes

This scheme has the following efficiency measures.

	Public Key (bits)	Private Key (bits)	Signature (bits)
Trapdoor [GPV08,MP12]	nmk	$2n^2k(1+\lceil \log_2(c) \rceil)$	$m \cdot (1+\lceil \log_2(s) \rceil)$

Table 4. GPV-Trapdoor storage requirements.

Scheme	(n,q)	Sizes [kB]		
		Public Key	Secret Key	Signature
GPV Ring	$(512, 2^{24})$	37.5	21.3	26.8
GPV Ring	$(512, 2^{27})$	47.3	23.9	30.1
TSS12 [Lyu12, Table 2] (based on decisional ring-LWE, m=2)	$(512, 2^{26})$	4.9	0.8	2.4
TSS12 [Lyu12, Table 2] (ring equivalent of l_2-SIS, m=17)	$(512, 2^{27})$	30.4	2.1	19.9
LyubSig [GLP12] (based on decisional ring-LWE, m=2)	$(512, 2^{23})$	1.4	0.2	1.1
BLISS I [DDLL13]	$(512, 2^{14})$	0.9	0.3	0.7
BLISS II [DDLL13]	$(512, 2^{14})$	0.9	0.3	0.6

Table 5. Comparison of our implementation with other lattice-based schemes with regard to storage sizes (in kilobytes).