MMSAT: A Scheme for Multimessage Multiuser Signature Aggregation

Yarkın Doröz[1], Jeffrey Hoffstein[2], Joseph H. Silverman[2], and Berk Sunar[1]

[1] Worcester Polytechnic Institute, Worcester MA, USA, {ydoroz,sunar}@wpi.edu
[2] Brown University, Providence RI, USA, {jhoff,jhs}@math.brown.edu

Abstract. Post-Quantum (PQ) signature schemes are known for large key and signature sizes, which may inhibit their deployment in real world applications. In this work, we construct a PQ signature scheme MMSAT that is the first such scheme capable of aggregating *unrelated messages* signed individually by different parties. Our proposal extends the notion of multisignatures, which are signatures that support aggregation of signatures on a single message signed by multiple parties. Multisignatures are especially useful in blockchain applications, where a transaction may be signed by multiple users. The proposed construction achieves significant gains in bandwidth and storage requirements by allowing aggregation of unrelated transactions. Our construction is derived by extending the $\mathsf{PASS}_{\mathtt{RS}}$ scheme, and thus the security of our scheme relies on the hardness of the Vandermonde-SIS problem. When aggregated, a signature consists of two parts. The first part is a *post-quantum size* signature that grows very slowly, scaling by on the order of $\log K$ bits for K signatures. The second part scales linearly with K, with a very short fixed cost, roughly twice the bit security. Thus even when aggregating a modest number of signatures, the per signature cost of MMSAT is in line with that of traditional pre-quantum signature schemes such as ECDSA. As an extension to MMSAT, we describe a variant called MMSATK in which it the public keys required to verify an aggregated signature are compressed by a factor of 20 to 30.

Keywords: Digital signature, PASS, signature aggregation, compressed key

Table of Contents

1 Introduction

Traditional cryptographic schemes providing encryption, key encapsulation, and signature services are expected to be replaced by quantum-resistant schemes in deployments during the next decade. The threat is so urgent that the US National Institute of Standards and Technology started a standardization competition in 2018 to select one or more so-called Post-Quantum schemes. In the meantime, companies such as Google have already started experimenting with PQ cryptography, including the *New Hope* scheme [5, 11] and an NTRU derivate [23, 25].

PQ Signature Schemes are expected to play a vital role in protecting the integrity of data in storage, during transmission, and even during computation [9, 19]. Indeed, in recent years there has been increasing interest in signature schemes having extended features. For example, *multisignatures* are a class of signature schemes that allow aggregation into a single signature of signatures produced by many parties on a single message [24]. Some multisignature schemes even support aggregation of the public keys, thereby greatly saving bandwidth and making them ideal for Blockchain applications [29], where transactions may be signed by multiple users and where reducing the block size is crucial in improving the throughput of the P2P network. Targeting the same application domain, a recent proposal by Bansarkhani and Sturm [7] proposed a PQ multisignature scheme. This is important, since the emergence of quantum computers will have a disastrous impact on current blockchain implementations. A significant feature offered by blockchains is that all past transactions are recorded on a ledger whose integrity is strongly protected, i.e., the ledger is expected to provide long-term security. The vast majority of Blockchains currently use traditional elliptic curve-based signatures (ECDSA), which is vulnerable to quantum attacks.

While techniques such as multisignatures are useful for compressing multiply signed individual transactions, the bulk of the transactions on Bitcoin and other networks are signed by different users. Therefore, new blocks are mostly made up of transactions with separate signatures that are not compressible by existing multisignature schemes. An early *aggregation scheme* capable of compressing independent messages signed by different users was proposed by Boneh et al. in [10]. The scheme uses bilinear maps constructed over a suitably chosen elliptic curve to achieve aggregation. Although at that time the primary proposed

application was to certificate chains, the aggregation scheme can be effectively used to compress signatures on transactions in a block. A more recent scheme by Boneh et al. [8] supports both signature compression and public-key aggregation with the primary goal being to reduce the size of the Bitcoin Blockchain. Their aggregation scheme is derived from Schnorr and BLS signatures. From an efficiency perspective, the methods in [8] appear to provide an ideal solution, since the scheme supports not only signature compression, but is also able to compress multiple public-keys into a single one. If deployed, an aggregation scheme such as [8] would greatly reduce the crypto overhead in Blockchains.

However, all of the aforementioned schemes use traditional cryptographic primitives that assume hardness in the traditional non-quantum model. There is an urgent need for PQ signature schemes that allow aggregation. This problem is exacerbated by the fact that most PQ schemes typically have signature and key sizes significantly larger than ECDSA, on the order of several thousand bits compared to the several hundred bits in ECDSA. For example, signatures in BLISS are 625 bytes, while those in ECDSA are 64 bytes [15]. Compression of individual PQ signatures and aggregation of public keys remain a challenge.

Our Contribution. The present work includes the following:

- We propose the first PQ signature scheme supporting aggregation across unrelated signatures signed by different users. It comes in several versions in which increased algorithm complexity yields improved operating characteristics. We denote these schemes by:

  $$\begin{aligned} \mathsf{MMSA} &= \text{Multi-message Multi-user Signature Aggregation Scheme,} \\ \mathsf{MMSAT} &= \mathsf{MMSA} \text{ with signature compression,} \\ \mathsf{MMSATK} &= \mathsf{MMSA} \text{ with both signature and public key compression.} \end{aligned}$$

  (The T in MMSAT refers to a linear map T used for signature compression.) An aggregated MMSAT signature has size roughly equal to a single PQ signature plus 2λ-bits per signature aggregated. From a practical perspective, even for a modest number of signatures, the aggregate signature size of MMSAT represents an improvement over traditional signature schemes such as ECDSA; e.g. it is 18-times smaller than BLISS and 1.9 times smaller than ECDSA for 1000 signatures at 128-bit security.
- We reduce the aggregate signature security of MMSAT to the forgery security of $\mathsf{PASS}^*_{\mathtt{RS}}$, a variant of $\mathsf{PASS}_{\mathtt{RS}}$.
- We use an ∞-norm analysis to give (slightly) improved estimates for the forgery probability from lattice reduction, leading to optimized parameters. This may be of independent interest for optimizing other lattice-based schemes.
- We analyze the lattice security of $\mathsf{MMSAT}$and provide concrete parameters secure against BKZ and Sieving.

Acknowledgements. The authors would like to thank Damien Stehlé and Thomas Prest for some very helpful discussions and comments on early drafts of this paper.

1.1 Comparision of MMSAT to Other Pre- and Post-Quantum Schemes

Table 1 on page 6 compares the parameter sizes of MMSAT and MMSATK with other post-quantum schemes such as BLISS and $\mathsf{PASS}_{\mathtt{RS}}$, as well as with ECDSA, which is not quantum secure. Since the other schemes do not support aggregation, we assume that an aggregate signature consists of K individual signatures. It is seen in the table that MMSAT provides superior performance against other post-quantum schemes, and indeed against ECDSA. Thus for an aggregate signature consisting of 10000 individual signatures at 128-bit security, MMSAT is 18 times smaller than BLISS, 70 times smaller than $\mathsf{PASS}_{\mathtt{RS}}$, and 1.9 times smaller than ECDSA. And at 256-bit security, MMSAT is 1.4 times smaller than in ECDSA. Thus MMSAT gives quantum-secure aggregate signatures that are roughly the same size as (but in fact smaller than) those provided by non-quantum-secure ECDSA. Further, as seen in the table, the size of individual public keys for MMSATK, which is our version of MMSAT with key compression, are only 5 to 8 times longer than ECDSA keys, and are 5 times shorter than the keys in post-quantum schemes such as BLISS.

The key sizes and signature sizes for MMSAT and MMSATK in Table 1 were computing using the formulas

$$\begin{aligned} \text{Public Key} &= t\log_2(q) \quad \text{bits}, \\ \text{Single Signature} &= N\log_2(b-k) + t\log_2(q) \quad \text{bits}, \\ \text{Aggregate Signature} &= N\log_2(B_k k\sqrt{K}) + t\log_2(q) + 2\lambda K \quad \text{bits}. \qquad (1) \\ \text{Compressed Key} &= 2\lambda + t'\log_2(q) \quad \text{bits} \qquad (2) \end{aligned}$$

We refer the reader to Table 2 for a list of the notation used in these formulas, to (7) for the derivation of (1) and to (9) for the derivation of (2).

2 Background, Notation, and the $\mathsf{PASS}_{\mathtt{RS}}$ Scheme

In this section we set notation and discuss the hard lattice problem underlying $\mathsf{PASS}_{\mathtt{RS}}$ and our various amalgamation schemes.

2.1 The SIS problem

The Short Integer Solution problem (SIS), introduced by Ajtai in 1996 [1], has been the basis of a number of cryptographic constructions. SIS is the problem of of finding a vector $\boldsymbol{y}$ in the kernel of a given linear transformation $\boldsymbol{A} : \mathbb{Z}_q^n \mapsto \mathbb{Z}_q^m$ such that $\boldsymbol{y}$ is small with respect to a prescribed norm. Ajtai showed a remarkable worst/average case equivalency property, i.e. for uniform random $\boldsymbol{A}$, for $m = \text{Poly}(n)$ with n also serving as the security parameter, the ability to solve random SIS instances with non-negligible probability implies an ability to find short vectors in any lattice within a certain approximation parameter. After Ajtai, a number of works [31,33,36] give better parameters that guarantee

Scheme	# of Signatures Aggregated	Uncompressed Public Key Size (Bytes)	Single Signature Size (Bytes)	Aggregate Signature Size (Bytes)	Compressed Public Key Size (Bytes)
	Security level: 128 bits				
MMSAT-128	1000	4266	2982	36	157
	5000	4647	3009	167	163
	10000	4828	3151	327	166
BLISS	1000			610	
Param II	5000	875	625	3052	875
	10000			6104	
$\mathsf{PASS}_{\mathtt{RS}}$	1000			2305	
Param 1153	5000	1500	2360	11523	1500
	10000			23046	
ECDSA	1000			63	
`nistp256`	5000	32	64	313	32
	10000			625	
	Security level: 256 bits				
MMSAT-256	1000	8257	5605	75	339
	5000	8904	5859	332	348
	10000	9209	5861	652	353
ECDSA	1000			94	
`nistp384`	5000	48	96	469	48
	10000			938	

Table 1: Size comparison of public keys, single signatures, and aggregate signatures, and compressed public keys at various aggregation levels

SIS hardness, while also improving the approximation factors for the underlying lattice problems. In [32] Micciancio and Peikert show that SIS retains its hardness even for small moduli $q \geq \beta n^{\delta}$ for any constant $\delta > 0$, where β is the bound on the Euclidean norm of the solution.

In practice, generic SIS instances do not yield compact and efficient implementations, so further assumptions are commonly made. An efficient class of schemes is obtained by replacing the matrix $\boldsymbol{A}$ with a list of elements $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k \in \mathbb{Z}_q[x]/\psi(x)$, where typically $\psi(x)$ is a cyclotomic polynomial. This gives rise to the Ideal-SISproblem (Ideal-SIS) of finding small $\boldsymbol{y}_i \in \mathbb{Z}_q[x]/\psi(x)$ satisfying $\sum \boldsymbol{a}_i \boldsymbol{y}_i = 0$. The security of the $\mathsf{PASS}_{\mathtt{RS}}$ scheme relies on a Vandermonde version of SIS(Vandermonde-SIS), in which $\boldsymbol{A}$ is a submatrix of the Fourier transform matrix, so $\boldsymbol{A}$ takes the form of a partial Vandermonde matrix. In the literature, Vandermonde-SIS is also referred to as the Partial Fourier Recovery Problem.

2.2 Some Notation and a Hard Problem

We fix:

N	a prime.
q	a prime satisfying $q \equiv 1 \pmod{N}$.
g	a primitive Nth root of unity in $\mathbb{Z}_q$.
R_q	the ring $\mathbb{Z}_q[x]/(x^N - 1)$, often identified with $\mathbb{Z}_q^N$.
$\star$	multiplication in R_q, equivalently, convolution product in $\mathbb{Z}_q^N$.
$\odot$	component-wise multiplication of vectors in $\mathbb{Z}_q^N$.

We often lift vectors in $\mathbb{Z}_q^N$ to vectors in $\mathbb{Z}^N$ with coordinates in $[-q/2, q/2]$.

Definition 1. *The* discrete Fourier transform over $\mathbb{Z}_q$ *is the linear transformation*

$$\mathcal{F} : \mathbb{Z}_q^N \longrightarrow \mathbb{Z}_q^N, \quad \boldsymbol{f} \longmapsto \mathcal{F}(\boldsymbol{f}) = \widehat{\boldsymbol{f}}, \quad \textit{given by the matrix} \quad (\mathcal{F})_{i,j} = g^{ij}.$$

Alternatively, as a map $R_q \to \mathbb{Z}_q^N$, the map $\mathcal{F}$ is given by the formula

$$\mathcal{F}(\boldsymbol{f}) = \big(\boldsymbol{f}(1), \boldsymbol{f}(g), \boldsymbol{f}(g^2), \ldots, \boldsymbol{f}(g^{N-1})\big).$$

The discrete Fourier transform is a ring isomorphism $(\mathbb{Z}_q^N, \star) \to (\mathbb{Z}_1^N, \odot)$,

$$\mathcal{F}(\boldsymbol{a} + \boldsymbol{b}) = \mathcal{F}(\boldsymbol{a}) + \mathcal{F}(\boldsymbol{b}) \quad \text{and} \quad \mathcal{F}(\boldsymbol{a} \star \boldsymbol{b}) = \mathcal{F}(\boldsymbol{a}) \odot \mathcal{F}(\boldsymbol{b}).$$

The hard problem underlying $\mathsf{PASS}_{\mathtt{RS}}$ is an underdetermined linear inversion problem that is sometimes called the *Partial Fourier Recovery Problem*. We fix a subset

$$\Omega = \{\Omega_1, \ldots, \Omega_t\} \subset \{0, 1, \ldots, N-1\},$$

and we use it to define a linear transformation $\mathcal{F}_\Omega$ that projects $\mathcal{F}(\boldsymbol{f})$ onto the coordinates specified by Ω. Thus $\mathcal{F}_\Omega : R_q \to \mathbb{Z}_q^t$ is the map

$$\mathcal{F}_\Omega(\boldsymbol{f}) = \widehat{\boldsymbol{f}}|_\Omega = \begin{pmatrix}\text{the coordinates of } \widehat{\boldsymbol{f}} \text{ that are} \\ \text{specified by the index set } \Omega\end{pmatrix} = \big(\boldsymbol{f}(g^i)\big)_{i \in \Omega}.$$

Definition 2. *The* Partial Fourier Recovery Problem *is:*

Given $\widehat{\boldsymbol{\kappa}}|_\Omega \in \mathbb{Z}_q^t$, find $\boldsymbol{\xi} \in R_q$ with small norm such that $\widehat{\boldsymbol{\xi}}|_\Omega = \widehat{\boldsymbol{\kappa}}|_\Omega$ in $\mathbb{Z}_q^t$.

Remark 1. The problem of recovering a signal from a restricted number of its Fourier coefficients is a well-studied problem that is known to be difficult in general. See Section 2.3 for a reformulation as a lattice problem. The original $\mathsf{PASS}_{\mathtt{RS}}$ paper [21] contains a more detailed discussion of the difficulty of solving the partial Fourier recovery problem.

Remark 2. $\mathsf{PASS}_{\mathtt{RS}}$ is an improved version of the original PASS scheme [22]. It uses a rejection sampling method pioneered by Lyubashevsky [27] to negate attacks based on transcript analysis. Lyubashevsky originally constructed a lattice-based transcript-secure identification scheme using a technique that he called "aborting," and which is often called *rejection sampling*. Lyubashevsky et al. later improved his technique and constructed a signature scheme via the Fiat–Shamir method, with hardness based on the Ring-SIS problem [15, 20, 28].

2.3 The Partial Fourier Recovery Problem as a Lattice Problem

We write the coordinates of $\widehat{\boldsymbol{\kappa}}|_\Omega$ as $\widehat{\kappa}_1, \ldots, \widehat{\kappa}_t$, and then the the Partial Fourier Problem asks for a short vector $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_{N-1}) \in \mathbb{Z}^N$ satisfying

$$\xi_0 + g^i \xi_1 + g^{2i} \xi_2 + \cdots + g^{(N-1)i} \xi_{N-1} \equiv \widehat{\kappa}_i \pmod{q} \quad \text{for all } i \in \Omega.$$

This gives t linear congruences in the N unknowns $\xi_0, \ldots, \xi_{N-1}$, and the fact that g is a primitive Nth root of unity in $\mathbb{Z}_q$ implies that the congruences are independent. (Indeed, the full set of congruences for $0 \le i < N$ is given by a vanderMonde matrix, which is invertible.) This means that we can solve for the first t unknowns in terms of the later ones. So for $0 \le k < t$, we have

$$\xi_k \equiv \sum_{j=t}^{N-1} h_{jk} \xi_j - c_k \pmod{q} \quad \text{with known } h_{jk}, c_k \in \mathbb{Z}_q.$$

Writing I_n for the n-by-n identity matrix, we consider the lattice

$$\Lambda = \operatorname{RowSpan} \begin{pmatrix} qI_t & 0 \\ H & I_{N-t} \end{pmatrix}, \quad \text{where} \quad H = \begin{pmatrix} h_{t0} & \cdots & h_{t,t-1} \\ \vdots & \ddots & \vdots \\ h_{N-1,0} & \cdots & h_{N-1,t-1} \end{pmatrix}. \tag{3}$$

Then Λ contains the vector

$$(\xi_0 + c_0, \ldots, \xi_{t-1} + c_{t-1}, \xi_t, \ldots, \xi_{N-1})$$

which is close to the known vector $(c_0, \ldots, c_{t-1}, 0, \ldots, 0)$, so the Partial Fourier Recovery Problem can be formulated as a Closest Vector Problem in the lattice Λ. And as usual, a CVP in dimension N can be solved by embedding it into an SVP in dimension $N+1$, so throughout this paper, we simply identify the CVP problem associated to partial Fourier recovery with an SVP problem in the same lattice.

Definition 3. *We define the* space of (N, q, t)-SIS lattices *to be set of lattices*

$$\Lambda_{N,q,t} = \left\{ \operatorname{RowSpan} \begin{pmatrix} qI_t & 0 \\ H & I_{N-t} \end{pmatrix} : H \in \operatorname{Mat}_{(N-t)\times t}(\mathbb{Z}_q) \right\}.$$

2.4 Further Notation

Elements $\boldsymbol{a} \in R_q$ are represented as polynomials

$$\boldsymbol{a} = a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1} \quad \text{with } a_i \in \mathbb{Z}_q,$$

or alternatively as a vector of coefficients

$$\boldsymbol{a} = [a_0, a_1, a_2, \ldots, a_{N-1}].$$

When convenient, we freely transition between these representations. For numerical calculations, we consistently identify $a_i \in \mathbb{Z}_q$ with an integer satisfying $|a_i| \le q/2$. With this convention, the L_∞-norm and L_2-norm of $\boldsymbol{a} \in R_q$ are

$$\|\boldsymbol{a}\|_\infty = \max_{0\le i<N} |a_i| \quad \text{and} \quad \|\boldsymbol{a}\|_2 = \sqrt{|a_0|^2 + \cdots + |a_{N-1}|^2}.$$

(The L_∞-norm is also called the *sup norm*.) We define sets of norm-bounded vectors and trinary vectors by

$$\mathcal{B}^\infty(k) = \big\{\boldsymbol{a} \in R_q : \|\boldsymbol{a}\|_\infty \le k\big\},$$
$$\mathcal{T}_N(d) = \left\{\begin{matrix}\text{polynomials in } R_q \text{ with } d \text{ coefficients equal to } 1,\\ d \text{ coefficients equal to } -1, \text{ and the rest equal to } 0\end{matrix}\right\}.$$

Finally, we write $[K]$ for the set $\{1, 2, \ldots, K\}$, and $\mathbb{1}_{\mathcal{S}}(x)$ for the indicator function that equals 1 if $x \in \mathcal{S}$ and 0 otherwise.

3 Description of $\mathsf{PASS}_{\mathtt{RS}}$

In this section we briefly describe a version of the $\mathsf{PASS}_{\mathtt{RS}}$ that is optimized for amalgamation. Further details about $\mathsf{PASS}_{\mathtt{RS}}$ may be found in the original paper [21]. The version of $\mathsf{PASS}_{\mathtt{RS}}$ that we describe here differs in small ways from the version in [21]; see Section 3.6 for details.

Table 2 gives a brief description of the public parameters used by $\mathsf{PASS}_{\mathtt{RS}}$, as well as additional parameters required for the amalgamation schemes MMSA, MMSAT, and MMSATK, which are the main topic of this paper.

3.1 Challenge Creation for $\mathsf{PASS}_{\mathtt{RS}}$ and MMSA Variants

In all versions of $\mathsf{PASS}_{\mathtt{RS}}$ and MMSA, the challenge polynomial $\boldsymbol{c} \in \mathcal{T}(d_c)$ is created by applying a hash function $\mathsf{Hash}_{\mathsf{C}}$ to data that includes the message digest μ and some quantities that depend on the public key $\widehat{\boldsymbol{f}}|_\Omega$ and the partial Fourier values $\widehat{\boldsymbol{y}}|_\Omega$ of the commitment. However, the different versions of $\mathsf{PASS}_{\mathtt{RS}}$ and MMSA feed slightly different quantities to $\mathsf{Hash}_{\mathsf{C}}$. For this reason, in all of these algorithms we will write simply

$$\boldsymbol{c} \leftarrow \mathsf{Hash}_{\mathsf{C}}(\mathsf{Scheme}, \mu). \tag{4}$$

The assignment (4) is an abbreviation for the assignments listed in Table 3.

3.2 $\mathsf{PASS}_{\mathtt{RS}}$ Key Generation

In the original formulation of $\mathsf{PASS}_{\mathtt{RS}}$ [22], the secret key was a randomly chosen polynomial $\boldsymbol{f} \in \mathcal{B}^\infty(1)$, so its coefficients were chosen independently and uniformly from $\{-1, 0, 1\}$. Here we add flexibility by taking $\boldsymbol{f}$ to be a randomly chosen element of $\mathcal{T}_N(d_f)$. The parameter d_f is selected so that that the key space has more than $2^{2\lambda}$ elements, and so that various lattice security estimates are satisfied. The public key corresponding to the secret key $\boldsymbol{f}$ is $\widehat{\boldsymbol{f}}|_\Omega = \mathcal{F}_\Omega(\boldsymbol{f})$, the partial Fourier transform of $\boldsymbol{f}$.

N	a prime (*dimension parameter*)
q	a prime satisfying $q \equiv 1 \pmod N$ (*modulus parameter*)
g	a primitive N^{th} root of unity in $\mathbb{Z}_q$
R_q	the ring $\mathbb{Z}_q[x]/(x^N - 1)$, often identified with $\mathbb{Z}_q^N$ with multiplication $\star$
$\star$	multiplication in R_q, convolution product in $\mathbb{Z}_q^N$
$\odot$	coordinate-by-coordinate multiplication in $R_q \cong \mathbb{Z}_q^N$
λ	bit security parameter
$\mathcal{M}$	space of message digests $\mu \in \mathcal{M}$
Ω	a subset of $\{g^j : 1 \le j \le N-1\}$
t	$= \lvert\Omega\rvert$, the number of elements in Ω
B_t	$\approx t/N$, parameter used to select t
t_0	dimension parameter for signature compression map T, satisfies $q^{t_0} \ge 2^{2\lambda}$
t'	dimension parameter for key compression, satisfies $q^{t'} \ge 2^{2\lambda}$ and $\binom{t}{t'} \ge 2^{2\lambda}$
k	L^∞-norm bound for commitment polynomial $\boldsymbol{y}$
b	L^∞-norm bound for rejection sampling is $k - b$
K	number of individualsignatures contained in an aggregate signature
B_k, B_q	multipliers used for L^1-norm bounds for aggregate signature, related by $B\sqrt{K}(k-b) \approx B_q q$.
d_c	the number of 1's and -1's in a challenge polynomial, $d_c \le b/2$
d_f	the number of 1's and -1's in a private key
T	a $\mathbb{Z}_q$-linear map $T : \mathbb{Z}_q^t \to \mathbb{Z}_q^{t_0}$ used for compression
$\mathsf{Hash_C}$	a hash/encoder function $\{0,1\}^* \to \mathcal{T}_N(d_c)$
$\mathsf{Hash_B}$	a hash/encoder function $\{0,1\}^* \to \{-1,1\}^K$
Hash_Ω	a hash/encoder function $\{0,1\}^* \to$ {subsets of Ω containing t' elements}

Table 2: Public parameters for $\mathsf{PASS}_{\mathtt{RS}}$, MMSA, and its variants

3.3 $\mathsf{PASS}_{\mathtt{RS}}$ Signing

Signing is an iteratative process consisting of the generation of a candidate signature followed by a rejection sampling step to prevent the publication of signatures that could leak secret key information.

The signer has a secret key $\boldsymbol{f}$ and wants to sign a message digest μ. She first selects a commitment polynomial $\boldsymbol{y}$ uniformly at random from $\mathcal{B}^\infty(k)$. The commitment $\boldsymbol{y}$ serves to mask the private key and must be treated with the same care as the private key itself. The signer then computes and stores the partial Fourier transform $\widehat{\boldsymbol{y}}|_\Omega = \mathcal{F}_\Omega \boldsymbol{y}$. The quantity $\widehat{\boldsymbol{y}}|_\Omega$ will ultimately be made public if the candidate signature passes rejection sampling.

Next, the signer computes a challenge polynomial $\boldsymbol{c} \in \mathcal{T}(d_c)$ that is used to bind $\widehat{\boldsymbol{y}}|_\Omega$ to μ. To do this she uses a public hash/encoder function:[3]

$$\mathsf{Hash_C} : \{0,1\}^* \to \mathcal{T}_N(d_c) \quad \text{to compute} \quad \boldsymbol{c} = \mathsf{Hash_C}(\widehat{\boldsymbol{y}}|_\Omega, \widehat{\boldsymbol{f}}|_\Omega, \mu).^4$$

[3] When we say that a function is a *hash/encoder function*, we mean that it has all of the standard properties of a cryptographically secure hash function such as SHA-512, but that its values lie in a set that may not be a simple set of bit strings $\{0,1\}^n$.

[4] More generally, as noted in Table 3, the input to $\mathsf{Hash_C}$ may instead include $T(\widehat{\boldsymbol{y}}|_\Omega)$ and/or $T(\widehat{\boldsymbol{f}}|_\Omega)$, depending on which variant of MMSA is being used.

$$\boldsymbol{c} \leftarrow \begin{cases} \mathsf{Hash}_{\mathsf{C}}(\widehat{\boldsymbol{y}}|_\Omega, \widehat{\boldsymbol{f}}|_\Omega, \mu) & \text{if Scheme} = \mathsf{PASS}_{\mathtt{RS}} \text{ or MMSA}, \\ \mathsf{Hash}_{\mathsf{C}}\big(T(\widehat{\boldsymbol{y}}|_\Omega), \widehat{\boldsymbol{f}}|_\Omega, \mu\big) & \text{if Scheme} = \mathsf{MMSAT}, \\ \mathsf{Hash}_{\mathsf{C}}\big(T(\widehat{\boldsymbol{y}}|_\Omega), T(\widehat{\boldsymbol{f}}|_\Omega), \mu\big) & \text{if Scheme} = \mathsf{MMSATK}. \end{cases}$$

Table 3: Input to $\mathsf{Hash}_{\mathsf{C}}$ for $\mathsf{PASS}_{\mathtt{RS}}$, MMSA, MMSAT, and MMSATK

The signer uses the commitment $\boldsymbol{y}$, the challenge $\boldsymbol{c}$, and her private key $\boldsymbol{f}$ to compute a candidate signature

$$\boldsymbol{z} = \boldsymbol{f} \star \boldsymbol{c} + \boldsymbol{y} \in R_q.$$

If $\|\boldsymbol{z}\|_\infty \leq k - b$, i.e., if every coefficient of $\boldsymbol{z}$ falls into the interval $[-k+b, k-b]$, then the signer outputs the signature $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu)$. Otherwise $\boldsymbol{y}$, $\boldsymbol{c}$, and $\boldsymbol{z}$ are discarded, and the signing process is repeated. It is proved in [21] that the $\boldsymbol{z}$ values of the signatures passing this rejection sampling procedure are uniformly distributed in $\mathcal{B}^\infty(k-b)$.

Remark 3. We note that even for the versions of MMSA that use the compression function T, the individual signatures always include the quantity $\widehat{\boldsymbol{y}}|_\Omega$, which the verifier uses in computing $\widehat{\boldsymbol{f}}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega + \widehat{\boldsymbol{y}}|_\Omega$. The only change is that the input to the hash function that creates the challenge polynomial $\boldsymbol{c}$ uses the compressed version $T(\widehat{\boldsymbol{y}}|_\Omega)$ of $\widehat{\boldsymbol{y}}|_\Omega$.

3.4 $\mathsf{PASS}_{\mathtt{RS}}$ Verification

To check the validity of the signature $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu)$ for the public key $\widehat{\boldsymbol{f}}|_\Omega$, the verifier first computes

$$\boldsymbol{c} = \mathsf{Hash}_{\mathsf{C}}(\widehat{\boldsymbol{y}}|_\Omega, \widehat{\boldsymbol{f}}|_\Omega, \mu).$$

The verifier accepts the signature as valid if

$$\boldsymbol{z} \in \mathcal{B}^\infty(k-b) \quad \text{and} \quad \widehat{\boldsymbol{z}}|_\Omega = \widehat{\boldsymbol{f}}|_\Omega \cdot \widehat{\boldsymbol{c}}|_\Omega + \widehat{\boldsymbol{y}}|_\Omega.$$

3.5 $\mathsf{PASS}_{\mathtt{RS}}$ Correctness

Since $\mathcal{F}_\Omega$ is a ring homomorphism that changes $\star$ multiplication to $\odot$ multiplication, we see that

$$\boldsymbol{z} = \boldsymbol{f} \star \boldsymbol{c} + \boldsymbol{y} \quad \text{implies that} \quad \widehat{\boldsymbol{z}}|_\Omega = \widehat{\boldsymbol{f}}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega + \widehat{\boldsymbol{y}}|_\Omega.$$

Hence any signature that comes out of the signing algorithm will be verified as valid by the verification algorithm.

3.6 Differences from the Original $\mathsf{PASS}_{\mathtt{RS}}$

The version of $\mathsf{PASS}_{\mathtt{RS}}$ that we use in this paper differs from the original version of $\mathsf{PASS}_{\mathtt{RS}}$ in [21] in three ways:

1. The original $\mathsf{PASS}_{\mathtt{RS}}$ signature was $(\boldsymbol{c}, \boldsymbol{z}, \mu)$, while we are using $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu)$. This is irrelevant from a security perspective, since the formulas

$$\boldsymbol{c} = \mathsf{Hash}_{\mathsf{C}}(T(\widehat{\boldsymbol{y}}|_\Omega), \mu, \widehat{\boldsymbol{f}}|_\Omega) \quad \text{and} \quad \widehat{\boldsymbol{y}}|_\Omega = \widehat{\boldsymbol{f}}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega - \widehat{\boldsymbol{z}}|_\Omega$$

   show that an attacker can pass from one form of the signature to the other using public knowledge. In practice, the vector $\boldsymbol{c}$ tends to require fewer bits to describe than $\widehat{\boldsymbol{y}}|_\Omega$, which is why one would use it for single signatures. However, our amalgamation algorithm MMSAT is going to require the $\widehat{\boldsymbol{y}}|_\Omega$ values, so it is easier (at least for the sake of exposition) if we take take the individual signatures to already be in the form $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu)$.
2. The original $\mathsf{PASS}_{\mathtt{RS}}$ scheme did not include the public key $\widehat{\boldsymbol{f}}|_\Omega$ as input to the hash function $\mathsf{Hash}_{\mathsf{C}}$ used to create the challenge polynomial $\boldsymbol{c}$. By including $\widehat{\boldsymbol{f}}|_\Omega$ or $T(\widehat{\boldsymbol{f}}|_\Omega)$ in the input to $\mathsf{Hash}_{\mathsf{C}}$, we tie $\boldsymbol{c}$ irrevocably to the key $\boldsymbol{f}$. This can only help the security of the scheme.
3. The original $\mathsf{PASS}_{\mathtt{RS}}$ scheme did not us a a linear transformation T. For individual $\mathsf{PASS}_{\mathtt{RS}}$ signatures, there is no reason to use T, but for the amalgamate signatures produced by MMSA, the map T can be used to compress the amalgamate signature and the associated public keys.

Algorithm 1 Sign

Input: $(\mu,\ \boldsymbol{f},\ \mathsf{Scheme})$
1: **repeat**
2: $\quad \boldsymbol{y} \xleftarrow{\$} \mathcal{B}^\infty(k)$
3: $\quad \boldsymbol{c} \leftarrow \mathsf{Hash}_{\mathsf{C}}(\mathsf{Scheme}, \mu)$
4: $\quad \boldsymbol{z} \leftarrow \boldsymbol{f} \star \boldsymbol{c} + \boldsymbol{y}$
5: **until** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$
Output: $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu)$

Algorithm 2 Verify

Input: $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu, \widehat{\boldsymbol{f}}|_\Omega, \mathsf{Scheme})$
1: $\boldsymbol{c} \leftarrow \mathsf{Hash}_{\mathsf{C}}(\mathsf{Scheme}, \mu)$
2: $\boldsymbol{Z} \leftarrow \widehat{\boldsymbol{f}}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega + \widehat{\boldsymbol{y}}|_\Omega$
3: **if** $\boldsymbol{z} \in \mathcal{B}^\infty(k-b)$ **and** $\boldsymbol{Z} = \widehat{\boldsymbol{z}}|_\Omega$ **then**
4: $\quad$ *result* $\leftarrow$ valid
5: **else**
6: $\quad$ *result* $\leftarrow$ invalid
7: **end if**
Output: *result*

Table 4: Sign and Verify Algorithms for $\mathsf{PASS}_{\mathtt{RS}}$ and MMSA[TK]. See Table 3 in Section 3.1 for the Input to $\mathsf{Hash}_{\mathsf{C}}$ for the Various Schemes

4 From $\mathsf{PASS}_{\mathtt{RS}}$ to MMSA: Aggregating Signatures

In this section we describe MMSA, which is our basic signature aggregation scheme. Later we describe modified versions of MMSA called MMSAT and MMSATK that allow compressed aggregated signatures and compressed public keys

4.1 Signature Aggregation

Suppose that K individual signers use their private keys $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_K$ to sign message digests $\mu_1, \ldots, \mu_K$. They do this by randomly choosing commitment polynomials $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K \in \mathcal{B}^\infty(k)$, using the hash function $\mathsf{Hash_C}$ to create their challenge polynomials $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$, and creating their signatures $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K \in \mathcal{B}^\infty(k-b)$. More precisely, the signature of individual i is the triple $(\widehat{\boldsymbol{y}}_i|_\Omega, \boldsymbol{z}_i, \mu_i)$ associated to the public key $\widehat{\boldsymbol{f}_i}|_\Omega$. A crucial aspect of the Fiat–Shamir construction is that the signer cannot influence the value of the challenge $\boldsymbol{c}_i$, since $\boldsymbol{c}_i$ is the output of a hash function. The verification of the i'th signature is via a proof of knowledge of the private key $\boldsymbol{f}_i$ and the fact, demonstrated via $\boldsymbol{c}_i$, that $\widehat{\boldsymbol{y}}_i|_\Omega$ really is a collection of values of a short polynomial $\boldsymbol{y}$. The goal in forming an aggregate signature is to have the verification of the aggregate signature imply, with high probability, the validity of all the individual signatures.

A crucial piece of the aggregate signature is a random linear combination of the $\boldsymbol{z}_i$. In order to determine which linear combination to use, the aggregator applies a hash/encoder function to the challenge polynomials and computes a vector $\boldsymbol{\beta}$ with ± 1 coordinates,[5]

$$\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K) = \mathsf{Hash_B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K) \in \{-1, 1\}^K.$$

Then one piece of the aggregate signature is the sum

$$\boldsymbol{z} = \sum_{i=1}^{K} \beta_i \boldsymbol{z}_i.$$

The aggregator checks that $\boldsymbol{z}$ satisfies[6]

$$\|\boldsymbol{z}\|_\infty \le B_k \sqrt{K}(k-b) \tag{5}$$

for an appropriately chosen public parameter B_k; see Section 4.2. The inequality (5) is then one of the conditions checked by the verifier.

Each coordinate of $\boldsymbol{z}$ is more-or-less a K-step random walk, with each step having length uniformly distributed in $[-k+b, k-b]$, which is why it is easy for the aggregator to construct $\boldsymbol{z}$ satisfying (5) for, say $B_k = 3$. It follows from (5)

[5] Since our $\mathsf{PASS}_{\mathtt{RS}}$ signatures have the form $(\widehat{\boldsymbol{y}}|_\Omega, \boldsymbol{z}, \mu)$, the aggregator must first recreate the $\boldsymbol{c}$ values via the formula $\boldsymbol{c} \leftarrow \mathsf{Hash_C}(\mathsf{Scheme}, \mu)$.

[6] If (5) fails, the aggregator can permute the order of the signatures being aggregated and try again, since this changes the value of $\boldsymbol{\beta}$. Alternatively, one might append a few bits to the signature and use them as a counter that is part of the input to $\mathsf{Hash_B}$.

that $\boldsymbol{z}$ can be stored in roughly $N \log_2(B_k k\sqrt{K})$ bits. The complete aggregate signature consists of $\boldsymbol{z}$ and the list of commitment values $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K$. Thus the length of an aggregate signature is approximately[7]

$$N \log_2(B_k k\sqrt{K}) + t \log_2(q) K \text{ bits.}$$

In Section 5 we will reduce this further by using a compression map T.

We note, however, that increasing B_k to make it easier for the aggregator will also make it easier for a forger to forge an aggregate signature. In particular, we must take $B_k\sqrt{K}(k-b)$ sufficiently small compared to q, so we define a new parameter B_q satisfying $B_k\sqrt{K}(k-b) \approx B_q q$, and we require that B_q be chosen from some suitable range, say $\frac{1}{4} \le B_q \le \frac{2}{5}$. On the other hand, we need b/k to be quite small, since the rejection rate for signatures is roughly $(1-b/k)^N$. Finally, the analysis of lattice-reduction attacks puts other constraints on the various parameters, especially q, N and t.

To verify the aggregated signature, the verifier first recomputes the $\boldsymbol{c}_i = \mathsf{Hash}_\mathsf{C}(\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$.[8] The verifier then uses the $\boldsymbol{c}_i$ to recompute the β_i via $\boldsymbol{\beta} = \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$. He then checks that

$$\widehat{\boldsymbol{z}}|_\Omega = \sum_{i=1}^{K} \beta_i \left(\widehat{\boldsymbol{f}_i}|_\Omega \odot \widehat{\boldsymbol{c}_i}|_\Omega + \widehat{\boldsymbol{y}_i}|_\Omega \right). \tag{6}$$

For a valid aggregate of valid signatures, the formula (6) is true, since $\boldsymbol{z} = \sum \beta_i \boldsymbol{z}_i$ and the partial Fourier transform is a ring homomorphism.

To summarize, an aggregate signature is valid if $\boldsymbol{z}$ is short and $\widehat{\boldsymbol{z}}|_\Omega$ takes on a value determined by the public keys $\widehat{\boldsymbol{f}_i}|_\Omega$, message digests μ_i, and commitments $\widehat{\boldsymbol{y}_i}|_\Omega$.

4.2 The Probability That an Aggregate Signature Is Accepted

Definition 4. *We use the standard notation* $\mathsf{erf}(x)$ *for the Gaussian error function*

$$\mathsf{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

Then a random variable $\mathcal{N}$ *that is normally distributed with finite mean* μ *and variance* σ^2 *satisfies*

$$\mathrm{Prob}\Big(|\mathcal{N} - \mu| \le C\sigma \Big) = \mathsf{erf}\left(\frac{C}{\sqrt{2}} \right).$$

[7] If one includes the message digests $\mu_1, \ldots, \mu_K$ in the signature, this adds roughly 2λ bits. And of course, the verifier also needs access to the public keys $\widehat{\boldsymbol{f}_1}|_\Omega, \ldots, \widehat{\boldsymbol{f}_K}|_\Omega$.

[8] We remark that $\widehat{\boldsymbol{f}_i}|_\Omega$ is included in the input to Hash_C in order to prevent a key forgery attack; see Section 8.2.

Algorithm 3 Aggregate

Input: $\left(\widehat{\boldsymbol{y}_i}|_\Omega, \boldsymbol{z}_i, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega\right)_{i\in[K]}$

for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{c}_i \leftarrow \mathsf{Hash}_\mathsf{C}(\widehat{\boldsymbol{y}}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$
end for
$\boldsymbol{\beta} \leftarrow \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\boldsymbol{z} \leftarrow \beta_1\boldsymbol{z}_1 + \cdots + \beta_K\boldsymbol{z}_K$
if $\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)$ **then**
  result $\leftarrow$ success
else
  result $\leftarrow$ failure
end if

Output: $(\boldsymbol{z}, \mu_i, \widehat{\boldsymbol{y}_i}|_\Omega)_{i\in[K]}$, *result*

Algorithm 4 VerifyAggregate

Input: $(\boldsymbol{z}, \mu_i, \widehat{\boldsymbol{y}_i}|_\Omega, \widehat{\boldsymbol{f}_i}|_\Omega)_{i\in[K]}$

for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{c}_i \leftarrow \mathsf{Hash}_\mathsf{C}(\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$
end for
$\boldsymbol{\beta} \leftarrow \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\boldsymbol{Z} \leftarrow \sum_{i=1}^{K}\left(\beta_i\left(\widehat{\boldsymbol{f}_i}|_\Omega \odot \widehat{\boldsymbol{c}_i}|_\Omega + \widehat{\boldsymbol{y}_i}|_\Omega\right)\right)$
if $\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)$ **and** $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$ are distinct **and** $\boldsymbol{Z} = \widehat{\boldsymbol{z}}|_\Omega$ **then**
  result $\leftarrow$ valid
else
  result $\leftarrow$ invalid
end if

Output: *result*

Table 5: MMSA: Aggregate Signature Algorithms

Theorem 1 (Central Limit Theorem). *Let $\mathcal{X}_1, \mathcal{X}_2, \ldots$ be independent identically distributed random variables with finite mean μ and variance σ^2. For $K \ge 1$, let $\mathcal{Z}_K = \mathcal{X}_1 + \cdots + \mathcal{X}_K$. Then*

$$\lim_{K\to\infty} \text{Prob}\left(|\mathcal{Z}_K - K\mu| \le C\sqrt{K}\sigma\right) = \mathsf{erf}\left(\frac{C}{\sqrt{2}}\right).$$

Proof. The random variable $\mathcal{Z}_K$ has mean $\mu(\mathcal{Z}_K) = K\mu$ and variance $\sigma^2(\mathcal{Z}_K) = K\sigma^2$, and the Central Limit Theorem [16, chapter VIII, section 4] says that the normalized random variable $\big(\mathcal{Z}_K - \mu(\mathcal{Z}_K)\big)/\sigma(\mathcal{Z}_K)$ approaches a normal distribution with mean 0 and variance 1 as $K \to \infty$.

Corollary 1. *The probability that the result of Algorithm* 3 *is* success*, i.e., the probability that an aggregate signature is accepted, is approximately given by*

$$\text{Prob(success)} = \text{Prob}\Big(\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)\Big) \approx \mathsf{erf}\left(\sqrt{3/2}B_k\right)^N.$$

Proof. We start with a particular coordinate, say the jth coordinate, of $\boldsymbol{z} = \sum \beta_i\boldsymbol{z}_i$. The jth coordinate of each $\boldsymbol{z}_i$ may be viewed as a random variable that is uniformly distributed in the interval $[-k+b, k-b]$.[9] Multiplying by $\beta_i \in \{\pm 1\}$

[9] More precisely, it is the sum of a random number in the interval $[-k, k]$ coming from the challenge $\boldsymbol{y}_i$ added to a value chosen from a generalized hypergeometric distribution coming from the product $\boldsymbol{f}_i \star \boldsymbol{c}_i$. However, as shown in [21], the rejection sampling process discards the non-uniform edge values in such a way that the remaining distribution is, as stated, uniform on $[-k+b, k-b]$.

yields the same distribution, so each coordinate of $\boldsymbol{z}$ is a sum of K independent and identically distributed random variables[10], each of which has norm $\mu = 0$ and variance $\sigma^2 = \frac{1}{3}(k-b)^2$.

If K is reasonably large, then we can use the central limit theorem (Theorem 1) to estimate the probability that the jth coordinate satisfies (5). We apply the theorem with $\mu = 0$, $\sigma^2 = \frac{1}{3}(k-b)^2$, and $C = \sqrt{3}B_k$ to obtain

$$\text{Prob}\Big(\big|j\text{th coordinate of } \boldsymbol{z}\big| \le B_k\sqrt{K}(k-b)\Big) \approx \mathsf{erf}\left(\sqrt{3/2}B_k\right).$$

The aggregator wants every coordinate of $\boldsymbol{z}$ to be in the interval $[-k+b, k-b]$, so treating the coordinates of $\boldsymbol{z}$ as independent random variables (which is almost, but not quite, true), we find that

$$\text{Prob}\Big(\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)\Big) \approx \mathsf{erf}\left(\sqrt{3/2}B_k\right)^N.$$

Table 6 gives some representative values. They indicate that $B_k = 3$ is a reasonable choice for a typical range of values fo N.

N	3000			5000			9000		
B	2.5	3	3.5	2.5	3	3.5	2.5	3	3.5
	95.628%	99.939%	100.00%	92.820%	99.898%	99.999%	87.448%	99.817%	99.999%

Table 6: Estimate $\mathsf{erf}\big(\sqrt{3/2}B\big)$ for the probability that the aggregate signature has norm $\le B\sqrt{K}(k-b)$ and is thus accepted

5 From MMSA to MMSAT: Compressing the Aggregate Signature

MMSA aggregates multi-message multi-user signatures, but the aggregate signature includes one $\widehat{\boldsymbol{y}}_i|_\Omega$ for each signature in the aggregate. To curb this growth, we introduce MMSAT, a variant of MMSA that significantly reduces the per signature overhead. We do this using a linear map

$$T : (\mathbb{Z}/q\mathbb{Z})^t \mapsto (\mathbb{Z}/q\mathbb{Z})^{t_0},$$

so $T(\widehat{\boldsymbol{y}}_i|_\Omega)$ requires only $t_0 \log_2(q)$ bits to describe. Choosing t_0 so that

$$t_0 \log_2(q) \approx 2^\lambda,$$

[10] To ease the calculation, we take a continuous distribution on the interval $[-k+b, k-b]$, rather than a discrete one. This means that we are ignoring some lower order terms.

we will achieve combinatorial security for $T(\widehat{\boldsymbol{y}}_i|_\Omega)$.

However, if the signature were to include only the $T(\widehat{\boldsymbol{y}}_i|_\Omega)$ values, then the validity check on $\boldsymbol{z}$ would involve only $T(\widehat{\boldsymbol{z}}|_\Omega)$, which would lead to a lattice attack in which the lattice discriminant is reduced from q^t to q^{t_0}. This would make it easier for a forger to find a sufficiently small vector, since the expected solution to the SVP would be smaller. To preclude this, we include some additional information in the signature; specifically, we include the quantity

$$\boldsymbol{Y} = \beta_1 \widehat{\boldsymbol{y}_1}|_\Omega + \cdots + \beta_K \widehat{\boldsymbol{y}_K}|_\Omega.$$

This allows the verifier to do two things:

1. The verifier can check that $T(\boldsymbol{Y})$ is equal to $\sum \beta_i T(\widehat{\boldsymbol{y}}_i|_\Omega)$, thereby ensuring that $\boldsymbol{Y}$ was correctly generated from $\widehat{\boldsymbol{y}_1}|_\Omega, \ldots, \widehat{\boldsymbol{y}_K}|_\Omega$.
2. The verifier can check that $\boldsymbol{z}$ has the correct value by comparing $\widehat{\boldsymbol{z}}|_\Omega$ with $\left(\sum \beta_i \widehat{\boldsymbol{f}}_i|_\Omega \odot \widehat{\boldsymbol{c}}_i|_\Omega\right) + \boldsymbol{Y}$. This verification takes place in $\mathbb{Z}_q^t$, not $\mathbb{Z}_q^{t_0}$, so the lattice used to create a forgery has dimension that depends on t.

We note that this explains why we modified $\mathsf{PASS}_{\mathtt{RS}}$ so that the input to $\mathsf{Hash}_{\mathsf{C}}$ includes $T(\widehat{\boldsymbol{y}}|_\Omega)$, instead of $\widehat{\boldsymbol{y}}|_\Omega$, since the person verifying the aggregate signature only has access to the $T(\widehat{\boldsymbol{y}}_i|_\Omega)$ values.

The MMSAT signature aggregation and verification algorithms are given by Algorithms 5 and 6 in Table 7.

An MMSAT aggregate of K signatures consists of one $\boldsymbol{z} \in \mathbb{Z}_q^N$, one $\boldsymbol{Y} \in \mathbb{Z}_q^t$, and K elements $T(\widehat{\boldsymbol{y}}_i|_\Omega) \in \mathbb{Z}_q^{t_0}$, so ignoring the message digests, the cryptographic part of the aggregate signature has length approximately

$$\underbrace{N\log_2(B_k k\sqrt{K})}_{\text{storage for } \boldsymbol{z}} + \underbrace{t\log_2(q)}_{\text{storage for } \boldsymbol{Y}} + \underbrace{Kt_0\log_2(q)}_{\text{storage for } T(\widehat{\boldsymbol{y}_1}|_\Omega), \ldots, T(\widehat{\boldsymbol{y}_K}|_\Omega)} \quad \text{bits.}$$

We may adjust t_0 so that $Z_q^{t_0}$ is too large to find a collision, so for bit security λ, we want $q^{t_0} \approx 2^{2\lambda}$. This means that the aggregate of K signatures has length approximately

$$N\log_2(B_k k\sqrt{K}) + t\log_2(q) + 2\lambda K \quad \text{bits.} \tag{7}$$

Thus an MMSAT aggregate signature has a fixed cost of $N\log_2(B_k k\sqrt{K}) + t\log_2(q)$, plus a short fixed cost of 2λ for each signature. In this sense, MMSAT is a post-quantum scheme that has scalability characteristics similar to those of ECDSA. Table 8 summarizes key and signature sizes for $\mathsf{PASS}_{\mathtt{RS}}$, MMSA, and MMSAT.

6 From MMSAT to MMSATK: Compressing the Public Key

In this section we describe a further variant of MMSA called MMSATK, in which both signatures and public keys are compressed. This variant may be useful in situtations in which one wants to store and transmit large numbers of public

Algorithm 5 Aggregate

Input: $\left(\widehat{\boldsymbol{y}_i}|_\Omega, \boldsymbol{z}_i, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega\right)_{i\in[K]}$

for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{c}_i \leftarrow \mathsf{Hash}_\mathsf{C}(T(\widehat{\boldsymbol{y}}|_\Omega), \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$
end for
$\boldsymbol{\beta} \leftarrow \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\boldsymbol{z} \leftarrow \beta_1\boldsymbol{z}_1 + \cdots + \beta_K\boldsymbol{z}_K$
$\boldsymbol{Y} \leftarrow \beta_1\widehat{\boldsymbol{y}_1}|_\Omega + \cdots + \beta_K\widehat{\boldsymbol{y}_K}|_\Omega$
if $\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)$ **then**
  result $\leftarrow$ success
else
  result $\leftarrow$ failure
end if

Output: $(\boldsymbol{z}, \boldsymbol{Y}, \mu_i, T(\widehat{\boldsymbol{y}_i}|_\Omega))_{i\in[K]}$, *result*

Algorithm 6 VerifyAggregate

Input: $\left(\boldsymbol{z}, \boldsymbol{Y}, \mu_i, T(\widehat{\boldsymbol{y}_i}|_\Omega), \widehat{\boldsymbol{f}_i}|_\Omega\right)_{i\in[K]}$

for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{c}_i \leftarrow \mathsf{Hash}_\mathsf{C}(T(\widehat{\boldsymbol{y}_i}|_\Omega), \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$
end for
$\boldsymbol{\beta} \leftarrow \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\boldsymbol{Z} \leftarrow \left(\sum_{i=1}^{K} \beta_i(\widehat{\boldsymbol{f}_i}|_\Omega \odot \widehat{\boldsymbol{c}_i}|_\Omega)\right) + \boldsymbol{Y}$
$\boldsymbol{W} \leftarrow \left(\sum_{i=1}^{K} \beta_i T(\widehat{\boldsymbol{y}_i}|_\Omega)\right)$
if $\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)$ **and** $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$ are distinct **and** $T(\boldsymbol{Y}) = \boldsymbol{W}$ **and** $\boldsymbol{Z} = \widehat{\boldsymbol{z}}|_\Omega$ **then**
  result $\leftarrow$ valid
else
  result $\leftarrow$ invalid
end if

Output: *result*

Table 7: MMSAT: Aggregate Signature Algorithms with Signature Compression

keys. The aggregation and verification algorithms for MMSATK are given by Algorithms 7 and 8 in Table 10 on page 21. To assist in understanding MMSATK, we describe the components of the aggregate signature; these are the new parameter and the the quantities that are output from Algorithm 7 and used as input to Algorithm 8.

Ω'	a t'-element subset of Ω chosen by a hash function, where t' is smaller than t, but large enough for combinatorial security.
$\boldsymbol{z} = \sum \beta_i \boldsymbol{z}_i$	aggregate of $\boldsymbol{z}$-part of signatures
$\boldsymbol{Y} = \sum \beta_i \widehat{\boldsymbol{y}_i}\|_\Omega$	aggregate of the commitments
μ_i	the ith message digest
$\boldsymbol{Y}_i = T(\widehat{\boldsymbol{y}_i}\|_\Omega)$	the ith Ω-Fourier commitment, compressed via T
$\boldsymbol{Y}'_i = \widehat{\boldsymbol{y}_i}\|_{\Omega'}$	the ith Ω'-Fourier commitment
$\boldsymbol{F}_i = T(\widehat{\boldsymbol{f}_i}\|_\Omega)$	the ith public key, compressed via T
$\boldsymbol{F}'_i = \widehat{\boldsymbol{f}_i}\|_{\Omega'}$	the ith Ω'-Fourier part of the public key

The ith compressed public key for the aggregate signature is the pair

$$(\boldsymbol{F}_i, \boldsymbol{F}'_i) \in \mathbb{Z}_q^{t_0} \times \mathbb{Z}_q^{t'}.$$

Scheme	Single Public Key	Single Signature	Aggregate Signature
$\mathsf{PASS}_{\mathtt{RS}}$	$t\log_2(q)$	$N\log_2(k)$	$KN\log_2(k)$
MMSA	$t\log_2(q)$	$N\log_2(k)$	$N\log_2(B_k\sqrt{K}k) + Kt\log_2(q)$
MMSAT	$t\log_2(q)$	$N\log_2(k)$	$N\log_2(B_k\sqrt{K}k) + t\log_2(q) + 2\lambda K$
MMSATK	$2\lambda + t'\log_2(q)$	$N\log_2(k)$	$N\log_2(B_k\sqrt{K}k) + t\log_2(q) + 2\lambda K$

Table 8: Public key and signature sizes in bits. Since $\mathsf{PASS}_{\mathtt{RS}}$ does not support aggregation, it aggregates K signatures by storing all of them.

The parameter t_0 is chosen so that $t_0\log_2(q) \approx 2\lambda$. The parameter t' is required to satisfy both

$$q^{t'} \geq 2^{2\lambda} \quad \text{and} \quad \binom{t}{t'} \geq 2^{2\lambda}. \tag{8}$$

The first condition comes from the fact that $\widehat{\boldsymbol{y}_i}|_{\Omega'}$ and $\widehat{\boldsymbol{f}_i}|_{\Omega'}$ are in $\mathbb{Z}_q^{t'}$, which has $q^{t'}$ elements, and we want to avoid collisions in their values. The second comes from the fact that the values Ω' of Hash_Ω are t'-element subsets of Ω, and we want the number of such subsets to be large enough so in any collection of 2^λ signatures, it is unlikely that any two of the signatures use the same Ω'. In practice it appears that the second condition in (8) is more stringent than the first.

In any case, the total size of the compressed public key is

$$\underset{\text{Key Length}}{\mathsf{MMSATK\ Public}} \approx (t_0 + t')\log_2(q) \text{ bits} \approx 2\lambda + t'\log_2(q) \text{ bits}. \tag{9}$$

Example 1. A sample parameter set from Table 19 has

$$(\lambda, N, q, t) = (128, 4111, 2^{24.909}, 1370).$$

The combinatorial condition for key compression security (8) shows that we may take $t' = 40$, since $\binom{1370}{40} \approx 2^{256.81}$. Then

$$\begin{aligned} \mathsf{MMSA\ Key\ Size} &\approx t\log_2(q) &&\approx 4266 \text{ Bytes}, \\ \mathsf{MMSATK\ Key\ Size} &\approx 2\lambda + t'\log_2(q) &&\approx 157 \text{ Bytes}. \end{aligned}$$

Further examples are given in Table 9, in which we compare the key sizes for MMSAT and MMSATK for a selection of parameter sets from Table 19. The value of t' in that table is chosen to ensure $\binom{t}{t'} \geq 2^{2\lambda}$. See Section 13 for a more detailed discussion on the choice of t'.

Remark 4. The subset Ω' of Ω is chosen in Step 9 of Algorithm 8 by computing a hash function whose input includes $\boldsymbol{Y}$ and $\boldsymbol{z}$. This means that a forger cannot predict in advance which coordinates of $\widehat{\boldsymbol{z}}|_\Omega$ will be used in the verification step, so if the Ω'-coordinates are correct, then there is a very high probability that all of the Ω-coordinates are correct. See Section 13 for a quantitative analysis.

Remark 5. The only place that $\boldsymbol{F}_i$, the ith public key compressed via T, is used in Algorithm 8 is in Step 2 as input to $\mathsf{Hash}_{\mathsf{C}}$, where it is used to recreate the challenge polynomial $\boldsymbol{c}_i$. This serves to tie $\boldsymbol{F}_i$ to $\boldsymbol{F}'_i$, the ith Ω'-Fourier part of the public key, via the computation in Step 5 of the sum involving the product $\boldsymbol{F}'_i \odot \widehat{\boldsymbol{c}}_i|_{\Omega'}$. And as noted in the previous remark, the identity involving the sum over the Ω'-part of the Fourier transform implies (with high probability) that the same identity holds for the full Ω-part of the Fourier transform, which in turn implies (with high probability) that the identity holds before Fourier transform is applied.

λ	B_t	p	K	N	q	t	t'	MMSAT Key Size (Bytes)	MMSATK Key Size (Bytes)
128	1/3	25%	1000	4111	$2^{24.909}$	1370	40	4266	157
128	1/3	25%	5000	4271	$2^{26.125}$	1423	40	4647	163
128	1/3	25%	10000	4349	$2^{26.651}$	1449	40	4828	166
256	1/3	25%	1000	7393	$2^{26.806}$	2464	82	8257	339
256	1/3	25%	5000	7643	$2^{27.965}$	2547	81	8904	348
256	1/3	25%	10000	7759	$2^{28.486}$	2586	81	9209	353

Table 9: Comparison of key sizes for MMSAT and MMSATK for various parameter sets from Table 19.

7 Transcript Security and Rejection Sampling

7.1 $\mathsf{PASS}_{\mathtt{RS}}$ Probability of Acceptance/Rejection

We estimate the probability that a potential signature is accepted by Step 5 of Algorithm 1. We see from our estimate that it is beneficial to take b as small as possible, since that's make it more likely that a generated signature will be accepted, but this is subject to the countervaling force from Theorem 2 that transcript security decreases as b decreases.

Proposition 1. *Assume $N \geq d_f \geq d_c$ and that $d_c \leq b \leq 2d_c$ and that $b+d_c \leq k$. (These conditions hold for any reasonable choice of $\mathsf{PASS}_{\mathtt{RS}}$ parameters.) Fix $\boldsymbol{f} \xleftarrow{\$} \mathcal{T}_N(d_f)$. Then*

$$\underset{\boldsymbol{y}\xleftarrow{\$}\mathcal{B}^\infty(k),\, \boldsymbol{c}\xleftarrow{\$}\mathcal{T}_N(d_c)}{\text{Prob}} \left(\|\boldsymbol{f}\star\boldsymbol{c}+\boldsymbol{y}\|_\infty \leq k-b\right) \gtrapprox (1-b/k)^N . \tag{10}$$

Proof. See [14, 21].

Algorithm 7 Aggregate

Input: $\left(\widehat{\boldsymbol{y}_i}|_\Omega, \boldsymbol{z}_i, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega\right)_{i\in[K]}$

for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{Y}_i \leftarrow T(\widehat{\boldsymbol{y}_i}|_\Omega)$
  $\boldsymbol{F}_i \leftarrow T(\widehat{\boldsymbol{f}_i}|_\Omega)$
  $\boldsymbol{c}_i \leftarrow \mathsf{Hash}_\mathsf{C}\big(\boldsymbol{Y}_i, \mu_i, \boldsymbol{F}_i\big)$
end for
$\boldsymbol{\beta} \leftarrow \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\boldsymbol{z} \leftarrow \beta_1\boldsymbol{z}_1 + \cdots + \beta_K\boldsymbol{z}_K$
$\boldsymbol{Y} \leftarrow \beta_1\widehat{\boldsymbol{y}_1}|_\Omega + \cdots + \beta_K\widehat{\boldsymbol{y}_K}|_\Omega$
$\Omega' \leftarrow \mathsf{Hash}_\Omega(\boldsymbol{z}, \boldsymbol{Y}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{F}'_i \leftarrow \widehat{\boldsymbol{f}_i}|_{\Omega'}$
  $\boldsymbol{Y}'_i \leftarrow \widehat{\boldsymbol{y}_i}|_{\Omega'}$
end for
if $\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)$ **then**
  result $\leftarrow$ success
else
  result $\leftarrow$ failure
end if

Output: $\left(\boldsymbol{z}, \boldsymbol{Y}, \mu_i, \boldsymbol{Y}_i, \boldsymbol{Y}'_i, \boldsymbol{F}_i, \boldsymbol{F}'_i\right)_{i\in[K]}$, *result*

Algorithm 8 VerifyAggregate

Input: $\left(\boldsymbol{z}, \boldsymbol{Y}, \mu_i, \boldsymbol{Y}_i, \boldsymbol{Y}'_i, \boldsymbol{F}_i, \boldsymbol{F}'_i\right)_{i\in[K]}$

for $i := 1$ **to** K **step** 1 **do**
  $\boldsymbol{c}_i \leftarrow \mathsf{Hash}_\mathsf{C}\big(\boldsymbol{Y}_i, \mu_i, \boldsymbol{F}_i\big)$
end for
$\boldsymbol{\beta} \leftarrow \mathsf{Hash}_\mathsf{B}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\Omega' \leftarrow \mathsf{Hash}_\Omega(\boldsymbol{z}, \boldsymbol{Y}, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_K)$
$\boldsymbol{Z}' \leftarrow \Big(\sum_{i=1}^{K} \beta_i(\boldsymbol{F}'_i \odot \widehat{\boldsymbol{c}_i}|_{\Omega'})\Big) + \boldsymbol{Y}|_{\Omega'}$
$\boldsymbol{W}' \leftarrow \sum_{i=1}^{K} \beta_i\boldsymbol{Y}'_i$
if $\|\boldsymbol{z}\|_\infty \le B_k\sqrt{K}(k-b)$ **and** $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K$ are distinct **and** $\boldsymbol{Z}' = \widehat{\boldsymbol{z}}|_{\Omega'}$ **and** $\boldsymbol{W}' = \boldsymbol{Y}|_{\Omega'}$ **then**
  result $\leftarrow$ valid
else
  result $\leftarrow$ invalid
end if

Output: *result*

Table 10: MMSATK: Aggregate Signature Algorithms with Signature and Public Key Compression

Remark 6. The paper [14], in addition to giving a derivation of the rough inequality (10), also contains a more accurate analysis showing that the probability in (10) is greater than

$$\left(1 - \frac{2b}{2k+1}\right)^N \operatorname*{Prob}_{\boldsymbol{c}\xleftarrow{\$}\mathcal{T}(d_c)} \left(\|\boldsymbol{f} \star \boldsymbol{c}\|_\infty \le b\right)^N. \tag{11}$$

The coefficients of $\boldsymbol{f} \star \boldsymbol{c}$ lie in the interval $[-2d_c, 2d_c]$, but they are highly clustered toward the middle of this interval. Hence as long as b isn't too much smaller than $2d_c$, the probability term in (11) will be very close to 1, and hence the value of (11) is very close to $(1-b/k)^N$. We also note that even if the probability estimate in Proposition 1 is slightly off, it only affects the efficiency of the signing operation. It has no effect on security, nor does it affect the verification or aggregation algorithms.

7.2 Absolute and Probabilistic Rejection Sampling

Rejection sampling is the test $\|\boldsymbol{z}\|_\infty \overset{?}{\leq} k - b$ performed in Step 5 of Algorithm 1 in Table 4. For appropriately chosen parameters, it ensures that a transcript of signatures leaks no information about the private key.

Proposition 2. *If* $\mathsf{PASS}_{\mathtt{RS}}$ *parameters are chosen to satisfy*

$$d_f \geq d_c \quad \textit{and} \quad b \geq 2d_c,$$

then the distribution of $\boldsymbol{z}$ *values in a transcript of signatures is independent of the private key used to create the signatures.*

Proof. This is proven in [21]. See also [14] for a more detailed explanation.

The transcript security provided by Proposition 2 comes at a cost. The parameter d_c needs to be large enough to ensure that the $\boldsymbol{c}$ sample space $\mathcal{T}_N(d_c)$ cannot be searched, so one wants $\#\mathcal{T}(d_c) \geq 2^{2\lambda}$. Then the requirement that $b \geq 2d_c$ makes it harder to sign, since it increases the likelihood that a candidate signature will be rejected in Step 5 of Algorithm 1.

An analysis of the proof of Proposition 2 shows that even if we relax the requirement $b \geq 2d_c$ a little bit, it is highly unlikely that $\boldsymbol{z}$-transcripts of practical length produced by different keys will be distinguishable. This is vague, but it is possible to develop a rigorous theory of probabilistic rejection sampling and probabilistic transcript security. See [14] for a complete description of this theory and its application to $\mathsf{PASS}_{\mathtt{RS}}$ in particular. For the present article, we are content to cite the following result from [14].

Theorem 2. *Let* $n \geq 1$ *and* $\alpha \geq 80$*, and let* (N, d_f, d_c, b) *be* $\mathsf{PASS}_{\mathtt{RS}}$ *parameters that satisfy the inequalty*[11]

$$\frac{2N}{\binom{N}{d_c, d_c}} \sum_{k+\ell \geq b} \binom{d_f}{k} \binom{d_f}{\ell} \binom{N - 2d_f}{d_c - k, d_c - \ell} \leq n^{-1} 2^{-\alpha - 1}.$$

Then a $\mathsf{PASS}_{\mathtt{RS}}$ *transcript containing at most* n *signatures gives less than a* $2^{-\alpha}$ *advantage in guessing which of two private keys was used in its generation.*

Proof. See [14].

8 Abstract Security of Aggregate Signatures

Boneh et al. in [10] introduced the notion of aggregate signatures and analyzed the security of such a scheme in the aggregate chosen-key security model. Briefly the aggregate chosen-key security model assumes an existential forgery adversary

[11] Here $\binom{r}{s,t}$ is the trinomial coefficient $r!/s!t!(r-s-t)!$. Alternatively, it is equal to the product of binomial coefficients $\binom{r}{s}\binom{r-s}{t}$.

$\mathcal{A}$ is given the public key of a victim. The adversary may use any other public key, and can even ask the victim to contribute authentic signatures on messages he choses. His goal is to craft an aggregate signature that will convince third party verifiers that the victim has contributed a signature on a message (other than the ones where he used the victim as a signing oracle). The advantage of $\mathcal{A}$ is defined as the success probability of the following game[12]:

Setup: The aggregate forger $\mathcal{A}$ is given a challenge public key $\widehat{\boldsymbol{f}_K}|_\Omega$. The forger $\mathcal{A}$ has no influence over how $\widehat{\boldsymbol{f}_K}|_\Omega$ was generated.

Queries: The forger $\mathcal{A}$ requests signatures with $\widehat{\boldsymbol{f}_K}|_\Omega$ on messages of his choice. The queries may be adaptively generated.

Response: The forger $\mathcal{A}$ returns a forged aggregate signature, along with the public keys $\widehat{\boldsymbol{f}_1}|_\Omega, \ldots, \widehat{\boldsymbol{f}_K}|_\Omega$ and messages $\mu_1, \mu_2, \ldots, \mu_K$. The additional $K-1$ public keys are chosen by the forger. The message μ_K may not be among the messages the victim has signed during the query stage.

Result: The forger $\mathcal{A}$ wins the game if the aggregate signature is a valid aggregate on messages $\mu_1, \mu_2, \ldots, \mu_K$ under keys $\widehat{\boldsymbol{f}_1}|_\Omega, \ldots, \widehat{\boldsymbol{f}_K}|_\Omega$ and $\mathcal{A}$ did not request a signature on μ_K under $\widehat{\boldsymbol{f}_K}|_\Omega$.

Definition 5 ([10]). *An existential signature forger $\mathcal{A}(\epsilon, N_S, N_H, K, \tau)$ that operates in the chosen key model runs in at most τ time, makes at most N_S queries to the signing oracle and at most N_H queries to the hash oracle, with at least ϵ advantage for an aggregate of at most K users. We call an aggregate signature scheme $(\epsilon, N_S, N_H, K, \tau)$-secure against existential forgery if no forger $\mathcal{A}(\epsilon, N_S, N_H, K, \tau)$ exists in the chosen key model.*

8.1 A Reduction of MMSA to $\mathsf{PASS}_{\mathtt{RS}}$

We are able to prove the following reduction from MMSA to $\mathsf{PASS}_{\mathtt{RS}}$.

Theorem 3. *Let $\mathsf{PASS}_{\mathtt{RS}}$ with parameters (q, N, k', b') be (ϵ', τ')-secure against existential forgery. Then MMSA with parameters (q, N, k, b) is $(\epsilon, N_S, N_H, \tau)$-secure against a chosen key forger for $k' = \mathcal{O}(1)k\sqrt{N_H}$ and $b' = \mathcal{O}(1)k\sqrt{N_H}$ and for all $\epsilon \geq N_H\epsilon'$ success probability and time*

$$\tau \leq \tau' - (2N_S+1)\tau_{\mathcal{F}_\Omega} - (4N_S+2N_H+3)\tau_\odot - (N_S+1)\tau_\times - (N_S+N_H)\tau_{\mathsf{H}} - \tau_{\mathsf{Hash}}$$

where $\tau_\times, \tau_\odot, \tau_{\mathcal{F}_\Omega}, \tau_{\mathsf{H}}$ and τ_{Hash}, denote time required for a polynomial multiplication, component-wise multiplication/addition, $\mathcal{F}_\Omega$ transformation, hash table insertion, and hash computation, respectively.

Proof. We postpone the proof to Section A.

[12] We have slightly rewritten the definition in [10] and adapted it to follow our notation.

8.2 Key Swap Attacks on Aggregate Signatures

A serious threat to multisignatures and signature aggregation schemes is the possibility of swapping the forger's key for the legitimate key. In this attack, the adversary does not conform to the usual key generation process, and instead choses public keys in way to gain advantage in forging aggregate signatures [30]. To demonstrate the vulnerability consider an aggregating scheme built on the $\mathsf{PASS}_{\mathtt{RS}}$ (single signature) signing and verification primitives where Alice has a legitimately generated public key $\widehat{\boldsymbol{f}_A}|_\Omega$. During verification Alice's public key is used to recover the public version of $\boldsymbol{y}$, i.e. $\widehat{\boldsymbol{y}}|_\Omega = \widehat{\boldsymbol{z}}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega$ as the key step to enable the check in the verification algorithm. In the aggregation scheme, again we use the same mechanism but this time simultaneously over multiple keys in the aggregated signature $\boldsymbol{z}$, i.e. $\sum \widehat{\boldsymbol{y}}|_\Omega = \widehat{\boldsymbol{z}}|_\Omega - \sum \widehat{\boldsymbol{f}_i}|_\Omega \odot \widehat{\boldsymbol{c}_i}|_\Omega$. For simplicity assume only two signatures are aggregated and the forger is contributing the second public key $\widehat{\boldsymbol{f}_B}|_\Omega$. A malicious party can craft a fake public key $\widehat{\boldsymbol{f}_B}|_\Omega = \widehat{\boldsymbol{f}}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega$ and eliminate the effect of Alice's key during aggregate verification by also forcing $\widehat{\boldsymbol{c}_A}|_\Omega = \widehat{\boldsymbol{c}_B}|_\Omega$. This is easily done in the original $\mathsf{PASS}_{\mathtt{RS}}$ scheme, since in that version the challenge polynomial is computed from $\mathsf{Hash}(\widehat{\boldsymbol{y}}|_\Omega, \mu)$ and thus does not depend on the public key. Note that the forger does not even know the actual secret key that yields the public key $\widehat{\boldsymbol{f}_B}|_\Omega$, since he does not know the short $\boldsymbol{f}_A$ related to $\widehat{\boldsymbol{f}_A}|_\Omega$, but this does not hinder the success of this aggregate forgery attack:

$$\begin{aligned} 2\widehat{\boldsymbol{y}}|_\Omega &= \widehat{\boldsymbol{z}}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega - \widehat{\boldsymbol{f}_B}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega \\ &= \widehat{\boldsymbol{z}}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega - (\widehat{\boldsymbol{f}}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega) \odot \widehat{\boldsymbol{c}}|_\Omega \\ &= \widehat{\boldsymbol{z}}|_\Omega - \widehat{\boldsymbol{f}}|_\Omega \odot \widehat{\boldsymbol{c}}|_\Omega \end{aligned}$$

The values $\widehat{\boldsymbol{f}_A}|_\Omega$ and $\widehat{\boldsymbol{y}}|_\Omega$ are completely decoupled, with Alice's public key eliminated from the right-hand side, although the verification primitive does use Alice's public key $\widehat{\boldsymbol{f}_A}|_\Omega$. Since the forger has the short version of $\widehat{\boldsymbol{f}}|_\Omega$ and also gets to choose the short $\boldsymbol{y}$, he can freely forge an aggregate signature.

To overcome this particularly strong threat, Micali et al. [30] propose using complex schemes such as Zero Knowledge Proofs that ensure the correct generation of the public keys. Boneh et al. in [10] chose a simpler route by requiring that the individual messages in an aggregate signature to be distinct.

In $\mathsf{PASS}_{\mathtt{RS}}$[13] and MMSAT we opt for a simple countermeasure by requiring the challenge polynomial $\boldsymbol{c}_i$ to be computed not only from the the commitment polynomial $\widehat{\boldsymbol{y}_i}|_\Omega$ and message μ_i, but also from the public key $\widehat{\boldsymbol{f}_i}|_\Omega$. To overcome this countermeasure, a key-forging adversary would have to find a hash collision, or a pseudo-collision with the linear sum of a number of colliding hashes, without significantly increasing the norm of the resulting vector.

[13] As noted earlier, the original version of $\mathsf{PASS}_{\mathtt{RS}}$ did not include the public key as the input to the hash function that generates $\boldsymbol{c}$, but as also noted earlier, including the public key cannot hurt the security of the scheme, and it only minimally affects efficiency.

9 Operating Characteristics of Lattice Reduction Algorithms

Before analyzing the various lattice attacks on $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT, we review some basic material on lattice reduction and give references to the relevant literature on the results achieved by lattice reduction algorithsm such as BKZ.

Definition 6 (Gram–Schmidt Orthogonalization Algorithm). *The* Gram-Schmidt orthogonalization *of a collection of linearly independent vectors* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N \in \mathbb{R}^n$ *is the list of vectors* $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_N^*$ *defined inductively by*

$$\boldsymbol{v}_1^* = \boldsymbol{v}_1 \quad \textit{and} \quad \boldsymbol{v}_i^* = \boldsymbol{v}_i - \sum_{j=1}^{i-1} \frac{\boldsymbol{v}_i \cdot \boldsymbol{v}_j}{\boldsymbol{v}_j \cdot \boldsymbol{v}_j} \boldsymbol{v}_j \quad \textit{for } i = 2, \ldots, N. \tag{12}$$

We note that the vectors $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_N^*$ *are pairwise orthogonal, but are not in general orthonormal.*

Definition 7. *Let* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ *be a basis of an* N*-dimensional lattice* L*. The* Hermite Root Factor (HRF) *is the quantity*

$$\delta = \delta(\boldsymbol{v}_1, L) = \left(\frac{\|\boldsymbol{v}_1\|_2}{\mathrm{Disc}(L)^{1/N}} \right)^{1/(N-1)}.$$

Definition 8 (Geometric Series Assumption). *Fix a blocksize* $\beta \geq 50$*. There is a constant* $\delta(\beta)$ *with the following property: Let* $L \subset \mathbb{R}^N$ *be a (random) lattice of dimension* $N \gg \beta$*. Let* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ *be the basis for* L *obtained as the output of some version of BKZ-*β*, and let* $\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_N^*$ *be the Gram-Schmidt reduction* (12) *of this basis. Then*

$$\|\boldsymbol{b}_i^*\|_2 \approx \delta(\beta)^{-2} \left\|\boldsymbol{b}_{i-1}^*\right\|_2 \quad \textit{for all } 1 \leq i \leq N.$$

Remark 7. As noted in [2, Definition 4], the Geometric Series Assumption as described in [37] takes a given β and heuristically determines the lengths of consecutive Gram–Schmidt basis vectors. It is reasonably accurate for blocksizes $\beta > 50$, provided that the blocksize is significantly smaller than the lattice dimension; cf. [13, 35, 41].

Proposition 3 (Experimental). *Fix a blocksize* $\beta \geq 50$*, and let* $L \subset \mathbb{R}^N$ *be a (random) lattice of dimension* $N \gg \beta$*. Then the HRF of the basis obtained by running BKZ-*β *on* L *is approximately equal to*

$$\delta_{\mathrm{BKZ}}(\beta) = \left(\frac{\beta}{2\pi e} \cdot (\beta\pi)^{1/\beta} \right)^{1/(2\beta-2)}. \tag{13}$$

Proof (Justification). Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ be the basis of L produced by BKZ-β. Repeatedly applying the geometric series assumption (Definition 8) gives

$$\|\boldsymbol{v}_i^*\|_2 \approx \delta(\beta)^{-2(i-1)} \|\boldsymbol{v}_1^*\|_2 .$$

This allows us to estimate

$$\mathrm{Disc}(L) = \prod_{i=1}^{N} \|\boldsymbol{v}_i^*\|_2 \approx \prod_{i=1}^{N} \Big(\delta(\beta)^{-2(i-1)} \, \|\boldsymbol{v}_1^*\|_2 \Big) = \delta(\beta)^{(N-1)N} \, \|\boldsymbol{v}_1^*\|_2^N \, .$$

Hence the $\delta(\beta)$ in Definition 8 satisfies

$$\delta(\beta) = \Big(\|\boldsymbol{v}_1^*\|_2 \,/\, \mathrm{Disc}(L)^{1/N} \Big)^{1/(N-1)},$$

so $\delta(\beta)$ is the HRF of the output obtained from running BKZ with blocksize β.

We now quote from [13]: "Experiments in [17] show that in practice, the Hermite root factor [for BKZ-β] is typically $\delta(\beta, n)$, where $\delta(\beta, n)$ converges rapidly as n grows to infinity for fixed β, and it is shown in [12] that the data supports the assumption that the HRF follows the asymptotic formula (13)."

Proposition 4 (Experimental/Extrapolated). *Fix a blocksize $\beta \geq 50$, and let $L \subset \mathbb{R}^N$ be a (random) lattice of dimension $N \gg \beta$. Then a conservative estimate for the cost of running BKZ-β on L (using a quantum computer) is*

$$\textit{Cost of BKZ-}\beta \approx 2^{0.265\beta}. \tag{14}$$

Thus in order to achieve bit security λ, it must be impossible to break the system by running BKZ-β with $\beta \leq \lambda/0.265$.

Proof (Justification). Various implementations suggest that the cost of running BKZ-β is roughly $2^{0.29\beta}$, although there would be nontrivial parallelism and memory issues in scaling such an implementation up to large blocksizes; cf. Section 9.1. On a sufficiently large quantum computer, i.e., one with an exponential (in β) amount of quantum RAM, it might be possible to reduce the runtime to around $2^{0.265\beta}$, so that is the highly conservative value that we have decided to use in this article.

9.1 A Note on the General Sieve Kernel

Recently more efficient algorithms for solving Short Vector Problem (SVP) have appeared in the literature. Especially sieving, which is asymptotically slower, yields better run times on real time experiments compared to popular SVP solving algorithms, i.e. BKZ-2.0. In [2], Ducas et. al shares experimental results of the sieving technique. From a practical point of view, the method achieves a 400-fold speedup compared to BKZ-2.0 for low dimensional lattices. This translates roughly to a 9-bit reduction in the security level. In order to estimate the security reduction for higher dimensional lattices, we also performed similar experiments using the G6K sieving software. However, it is hard to solve SVP on lattices with dimensions over 120 in practice. In [2], the authors note that the performance observed for low dimensional lattices does not match with the asymptotic analysis, and they therefore refrain from extrapolating to higher dimensions. As

it stands now, while useful, using low dimensional G6K runtime estimates to extrapolate the runtime for higher lattice dimensions does not appear to be a viable option.

The authors of [2] also suggest a conservative security bound of $2^{0.292d}$ for solving the SVP in dimension d. Assuming this estimate, this gives a security bound of $2^{0.292d}$ for finding an entire KZ-reduced basis in dimension d, since the first element of such a basis is a solution to SVP. We note that since BKZ-β needs to find many KZ-reduced bases for many sublattices of dimension β, the SVP security estimate in [2] says that the security of BKZ-β is at least $2^{0.292\beta}$. (And presmuably considerably higher.)

In conclusion: Current conservative estiamtes for the practical security of BKZ-β on a classical computer suggest that it has bit security greater than 0.292β. This is far greater than the highly conservative quantum bit security of 0.265β for BKZ-β that we use in this paper.

9.2 The BKZ-Simulator HRF versus the HRF Formula

Formula (13) gives an estimate for the HRF of BKZ-β that is experimentally reasonably accurate provided that the lattice dimension is significantly larger than the blocksize β. An alternative way of estimating this HRF is the BKZ-Simulator of Chen and Nguyen [13, Algorithm 2]. They analyzed state-of-the-art BKZ lattice reduction algorithms and devised a BKZ-Simulator that predicts the quality of the output from running BKZ for a given number of rounds using a given, reasonably large, blocksize. See Section C for a detailed description of the BKZ-Simulator and our implementation.

We ran the BKZ-Simulator for 30 rounds[14] on $\mathsf{PASS}_{\mathtt{RS}}$ lattices i.e., on an (N, q, t)-SIS lattice as described in Definition 3. This is particularly easy, since the public lattice (3) for $\mathsf{PASS}_{\mathtt{RS}}$ is already in Hermite normal form. Then Gram–Schmidt reduction of the $\mathsf{PASS}_{\mathtt{RS}}$ lattice gives the following list of logarithmic lengths, which are used as input to the BKZ-Simulator:

$$(\underbrace{\log_2(q), \ldots, \log_2(q)}_{t \text{ copies of } \log_2(q)}, \underbrace{0, 0, 0, \ldots, 0}_{N-t \text{ copies of } 0}).$$

Table 11 compares the HRF output from the BKZ-Simulator to the HRF from the formula. Each section of the table gives three blocksizes of decreasing dimension-to-blocksize ratio. We can summarize the conclusions from the table as follows:

Dimension to Blocksize Ratio	**Formula HRF versus Simulator HRF**
Larger	HRF-Formula $>$ HRF-Simulator
Medium	HRF-Formula $\approx$ HRF-Simulator
Smaller	HRF-Formula $<$ HRF-Simulator

[14] The output from the simulator tends to stabilize after about 20 rounds, so running additional rounds would not improve the output.

As the dimension-to-blocksize ratio decreases, one expects the HRF-formula to become less accurate. On the other hand, for the larger dimension-to-blocksize ratios in Table 11, we expect that the HRF-formula should be farily accurate, which raises the question of why the BKZ-Simulator gives somewhat smaller HRF values in these cases. We suspect that the answer lies in the fact that a standard basis for a $\mathsf{PASS}_{\mathsf{RS}}$ (or NTRU) lattice is already in Hermite normal form, and the associated Gram-Schmidt basis has the special form

$$\{q\boldsymbol{e}_1, q\boldsymbol{e}_2, \ldots, q\boldsymbol{e}_t, \boldsymbol{e}_{t+1}, \boldsymbol{e}_{t+1}, \ldots, \boldsymbol{e}_N\}.$$

The BKZ-Simulator seems to be able to exploit special Gram-Schmidt bases of this sort, although running actual implementions of BKZ on $\mathsf{PASS}_{\mathsf{RS}}$ and NTRU lattices does not appear to yield significantly better results than running on random lattices.

This appears also to be the conclusion reached in [13], where the authors state: "But this model [used in the BKZ simulator] may not work with bases of special structure such as partial reductions of the NTRU Hermite normal form, which is why we only consider random reduced bases as input."

Based on these considerations, in all of our security calculations we have used the HRF formula (13) for the output of BKZ-β, together with the very conservative formula (14) for the effort required to run BKZ-β.

Blocksize	Bit Security	N	q	t	HRF from formula	HRF from BKZ-Simulator	Dimension / Blocksize
484	128.26	5297	$2^{35.544}$	1765	1.003484	1.003107	10.94
884	234.26	5297	$2^{35.544}$	1765	1.002242	1.002264	5.99
1284	340.26	5297	$2^{35.544}$	1765	1.001687	1.001704	4.13
967	256.25	9311	$2^{35.305}$	3103	1.002096	1.001754	9.63
1267	335.75	9311	$2^{35.305}$	3103	1.001705	1.001695	7.35
1567	415.25	9311	$2^{35.305}$	3103	1.001446	1.001455	5.94
484	128.26	3343	$2^{25.871}$	1671	1.003484	1.002687	6.91
884	234.26	3343	$2^{25.871}$	1671	1.002242	1.002270	3.78
1284	340.26	3343	$2^{25.871}$	1671	1.001687	1.001719	2.60
967	256.25	6007	$2^{27.717}$	3003	1.002096	1.001601	6.21
1267	335.75	6007	$2^{27.717}$	3003	1.001705	1.001601	4.74
1567	415.25	6007	$2^{27.717}$	3003	1.001446	1.001461	3.83

Table 11: Comparison of the HRF from the BKZ-Simulator and the Blocksize Formula (13) in Proposition 3

9.3 L_2-Norm Bounds Versus L_∞-Norm Bounds

In this section we let L be a lattice of dimension N and discriminant D. Suppose that the forger is required to find a vector $\boldsymbol{z} \in L$ satisfying $\|\boldsymbol{v}\|_\infty \le C$. The triangle inequality gives the trivial estimate

$$\|\boldsymbol{v}\|_2 \le \sqrt{N} \cdot \|\boldsymbol{v}\|_\infty ,$$

with equality if and only if all of the coordinates of $\boldsymbol{v}$ have the same magnitude. In particular, a necessary condition for success is that the vector $\boldsymbol{z} \in L$ satisfy

$$\|\boldsymbol{v}\|_2 \le \sqrt{N} \cdot C. \tag{15}$$

However, this is far from sufficient, since it is quite unlikely that the coordinates of a random $\boldsymbol{v}$ will all be of the same size.

We can give an upper bound for the probability that a random point $\boldsymbol{v}$ with L_2 norm $\|\boldsymbol{v}\|_2 \le R$ has L_∞ norm bounded by $\|\boldsymbol{v}\|_\infty \le C$ by calculating the following ratio of volumes:

$$\begin{aligned}
\operatorname*{Prob}_{\boldsymbol{v} \leftarrow \{\boldsymbol{w} \in \mathbb{R}^N : \|\boldsymbol{w}\|_2 \le R\}} \big(\|\boldsymbol{v}\|_\infty \le C\big) &\approx \frac{\operatorname{Vol}\big(\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\|_\infty \le C \text{ and } \|\boldsymbol{v}\|_2 \le R\}\big)}{\operatorname{Vol}\big(\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\|_2 \le R\}\big)} \\
&\le \frac{\operatorname{Vol}\big(\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\|_\infty \le C\}\big)}{\operatorname{Vol}\big(\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\|_2 \le R\}\big)} \qquad (16) \\
&= \frac{(2C)^N}{R^N \mu_N}. \qquad (17)
\end{aligned}$$

Suppose that the attacker runs BKZ-β on L. We denote the run-time of BKZ-β by Time_β and the HRF of the output by δ_β. The definition of HRF says that the output is a lattice vector $\boldsymbol{z}$ whose L_2-norm is approximately

$$\|\boldsymbol{z}\|_2 \approx \delta_\beta^{N-1} \cdot D^{1/N}. \tag{18}$$

An upper bound for the probability that such a vector has L_∞-norm smaller than C is given by (17). Hence the (probable) run-time to find a $\boldsymbol{z} \in L$ satisfying $\|\boldsymbol{z}\|_\infty \le C$ is

$$\begin{aligned}
\text{Time}_{\text{total}} &= \frac{\text{Run-time for BKZ-}\beta}{\text{Prob(BKZ-}\beta\text{ output satisfies } \|\boldsymbol{z}\|_\infty \le C)} \\
&= \frac{\text{Time}_\beta}{\operatorname{Prob}\big(\|\boldsymbol{z}\|_\infty \le C \;\big|\; \|\boldsymbol{z}\|_2 = \delta_\beta^{N-1} \cdot D^{1/N}\big)} \quad \text{from (18)}, \\
&\ge \frac{\text{Time}_\beta}{\big(2C/(\delta_\beta^{N-1} \cdot D^{1/N})\big)^N \cdot \mu_N^{-1}} \quad \text{from (17)}.
\end{aligned}$$

Taking logs and doing a little algebra yields

$$\log_2(\text{Time}_{\text{total}}) \ge \log_2(\text{Time}_\beta) + N\Bigg\{ \log_2\Big(\underbrace{\frac{\delta_\beta^{N-1} \cdot D^{1/N}}{\sqrt{N}C}}_{\substack{\text{must be} \le 1 \\ \text{from (15) and (18)}}}\Big) + \log_2\Big(\underbrace{\frac{1}{2}\mu_N^{1/N} \cdot \sqrt{N}}_{\approx\sqrt{\pi e/2}}\Big)\Bigg\}. \tag{19}$$

(The second subnote uses $\mu_N = \pi^{N/2}/\Gamma(1+N/2) \approx (2\pi e/N)^{N/2}$.)

We note that (19) gives a lower bound for the logarithmic-run-time, while the second term in the lower bound is a multiple of $\overline{N}$. So the attacker's total run-time will be huge unless the quantity in braces is non-positive. Hence to have any chance of success, the forger must choose β so that

$$\delta_\beta^{N-1} \cdot D^{1/N} \le \left(\frac{1}{2}\mu_N^{1/N} \cdot \sqrt{N}\right)^{-1} \sqrt{N}C.$$

Comparing this with (15) and (18), we see that the actual L_2-norm required to forge is roughly $\sqrt{2/\pi e}$ times smaller than the naive estimate (15) coming from the (implausible) assumption that the coordinates of a random vector are all of the same size. Hence in order to find a vector whose L_∞-norm is less than C, i.e., to find a forgery, the forger needs to achieve an HRF that is at most

$$\left(\left(\frac{1}{2}\mu_N^{1/N} \cdot \sqrt{N}\right)^{-1} \cdot \frac{\sqrt{N}C}{D^{1/N}}\right)^{1/(N-1)} = \left(\frac{2C}{\mu_N^{1/N} \cdot D^{1/N}}\right)^{1/(N-1)}. \tag{20}$$

Thus the forger needs to run BKZ-β with a sufficiently large blocksize β so that δ_β is at most the quantity given in (20).

Remark 8. We note that since $2\Gamma(1+N/2)^{1/N}/\sqrt{\pi} \approx \sqrt{2/\pi e}\cdot\sqrt{N}$, the attacker's HRF forgery goal is roughly $(2/\pi e)^{1/(2N-2)}$ times harder than suggested using the naive L_2-norm bound.

We summarize the preceding discussion as a proposition.

Proposition 5. *Let L be a lattice of dimension N. To find a vector $\boldsymbol{v} \in L$ whose L_∞-norm satisfies $\|\boldsymbol{v}\|_\infty \le C$, a lattice reduction algorithm needs to achieve a Hermite root factor δ satisfying*

$$\delta \le \left(\frac{2C\Gamma(1+N/2)^{1/N}}{\sqrt{\pi}\,\mathrm{Disc}(L)^{1/N}}\right)^{1/(N-1)}.$$

Proof. This is a restatement of (20).

Experimental Estimates for L_2-Norm Versus L_∞-Norm The inequality (16) is only an estimate since for any given C and R, a portion of the L_∞-box may lie outside the L_2-ball.

We start by observing that the conditional probability

$$\mathrm{Prob}\Big(\|\boldsymbol{v}\|_\infty \le C \;\Big|\; \|\boldsymbol{v}\|_2 \le R\Big)$$

is homogeneous, i.e., it only depends on the ratio C/R, so we study its value for $R = 1$. Then formula (17) says that

$$\mathrm{Prob}\Big(\|\boldsymbol{v}\|_\infty \le C \;\Big|\; \|\boldsymbol{v}\|_2 \le 1\Big) \le (2^N/\mu_N) \cdot C^N.$$

And of course, the probability is always at most 1, so this estimate is non-trivial for $C \le \frac{1}{2}\mu_N^{1/N}$. We are going to use this inequality and the following elementary proposition to estimate the mean value of $\|\boldsymbol{v}\|_\infty$.

Proposition 6. *Let $\mathcal{X}$ be a (well-behaved) random variable with values in $[0,1]$, and suppose that the distribution function $F_\mathcal{X}$ of $\mathcal{X}$ satisfies*

$$F_\mathcal{X}(x) \le Ax^N \quad \textit{for all } x \le A^{-1/N}.$$

Then the expectation of $\mathcal{X}$ satisfies

$$\mathrm{Mean}(\mathcal{X}) \ge \frac{NA^{-1/N}}{N+1}.$$

Proof. We compute

$$\begin{aligned}\mathrm{Mean}(\mathcal{X}) &= \int_0^1 xF'_\mathcal{X}(x)\,dx = xF_\mathcal{X}(x)\Big|_0^1 - \int_0^1 F_\mathcal{X}(x)\,dx \\ &= 1 - \int_0^1 F_\mathcal{X}(x)\,dx \ge 1 - \int_0^{A^{-1/N}} Ax^N\,dx - \int_{A^{-1/N}}^1 1\,dx = \frac{NA^{-1/N}}{N+1}.\end{aligned}$$

Applying Proposition 6 to our situation with $A = 2^N/\mu_N$, we find that

$$\operatorname*{Mean}_{\|\boldsymbol{v}\|_2\le 1}\bigl(\|\boldsymbol{v}\|_\infty\bigr) \ge \frac{N\mu_N^{1/N}}{2(N+1)} \approx \sqrt{\frac{\pi e}{2N}}. \tag{21}$$

In order to gauge the accuracy of the estimate (21), we experimented by choosing a large number of random points on the surface of an N-dimensional unit ball[15] and computed the mean value of the L_∞-norms of the points. As shown in Table 12, the theoretical lower bound (21) is a bit smaller than the experimental value, but the order of magnitude is about right.

Remark 9. There are well-known methods to generate points uniformly distributed on a sphere; see [34,39] We used the following algorithm. Let $\mathcal{X}_1, \ldots, \mathcal{X}_N$ be independent random variables that are normally distributed with mean 0 and variance 1. Then the random variable

$$\mathcal{S}_N = \frac{(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N)}{\sqrt{\mathcal{X}_1^2 + \mathcal{X}_2^2 + \cdots + \mathcal{X}_N^2}}$$

gives points that are uniformly distributed on the unit $(N-1)$-sphere in $\mathbb{R}^N$.

[15] Choosing points at random on the surface of the ball is the conservative choice, since we're not sure to what extent an algorithm such as BKZ will find points with L_2-norm significantly smaller than the expected norm. In any case, the majority of the volume of an N-ball lies near its surface, e.g., if $N \ge 1000$, then more than 99% of the volume lies within 0.005 of the surface.

N	$\frac{N\mu_N^{1/N}}{2(N+1)}$ from (21)	$\mathrm{Mean}\big(\|\mathcal{S}_N\|_\infty\big)$ based on $8 \cdot 10^6$ samples
1000	0.065017	0.108662
3000	0.037656	0.067998
4000	0.029189	0.060029

Table 12: Comparing the theoretical lower bound for $\mathrm{Mean}\big(\|\boldsymbol{v}\|_\infty\big)$ in the unit ball to the experimental mean on the unit sphere

9.4 The Gaussian Heuristic

Definition 9. *We denote the volume of a unit ball in $\mathbb{R}^N$ by*

$$\mu_N = \mathrm{Vol}\Big(\big\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\|_2 \leq 1\big\}\Big) = \frac{\pi^{N/2}}{\Gamma(1+N/2)} \approx \left(\frac{2\pi e}{N}\right)^{N/2},$$

where the approximation comes from Stirling's formula $\Gamma(1+x) \sim (x/e)^x$ and is valid for large values of N. In general, the volume of a ball of radius R in $\mathbb{R}^N$ is[16]

$$\mathrm{Vol}\Big(\big\{\boldsymbol{v} \in \mathbb{R}^N : \|\boldsymbol{v}\|_2 \leq R\big\}\Big) = R^N \mu_N.$$

Definition 10 (Gaussian Heuristic Assumption). *The* Gaussian Heuristic Assumption *says that a smallest non-zero vector in a random N-dimensional lattice Λ has L_2-norm approximately equal to the* Gaussian Heuristic Value

$$\mathsf{GH}(\Lambda) = \frac{\mathrm{Disc}(\Lambda)^{1/N}}{\mu_N^{1/N}} \approx \sqrt{\frac{N}{2\pi e}} \cdot \mathrm{Disc}(\Lambda)^{1/N}. \tag{22}$$

More generally, for $\epsilon > 0$, it is expected that there are no non-zero vectors of length $(1-\epsilon)\,\mathsf{GH}(\Lambda)$, and that there are lots of vectors of length $(1+\epsilon)\,\mathsf{GH}(\Lambda)$.

Remark 10. The justification for the Gaussian Heuristic Assumption is that the volume of a ball of radius $\mathsf{GH}(\Lambda)$ is equal to the volume of a fundamental domain for Λ.

9.5 The Unique Shortest Vector Problem (UniqueSVP)

Definition 11. *The* Unique Shortest Vector Problem (UniqueSVP) *refers to finding a shortest vector in a lattice whose shortest vector is significantly shorter than the next smallest linearly independent vector. The difficulty of the* UniqueSVP *is measured by the ratio of the first and second successive minima of the lattice. Since in practice it is difficult to compute the second minimum, we approximate it by the Gaussian heuristic value, so we define the* SVP-Gap *of the lattice L to be the quantity*

$$\mathsf{SVP\_Gap}(L) = \frac{\mathsf{GH}(L)}{\lambda_1(L)} \approx \frac{\lambda_2(L)}{\lambda_1(L)}.$$

[16] See [40] for a nice exposition of several ways to compute the volumes of the N-ball.

There are two esimtates for the difficulty of solving UniqueSVP. The first, which may be found in [3, 17, 18], is based on analysis of experiments using various classes of lattices. The second based on an observation in Alkim–Ducas–Pöppelmann–Schwabe [6] that one should declare success in solving UniqueSVP as soon as the intermediate results of a lattice reduction algortihm such as BKZ deviates from the expected intermediate results for a lattice not containing an especially short vector. In other words, declare success when BKZ solves the decisional UniqueSVP problem. This second approach is analyzed in more detail in Albrecht–Göpfert–Virdia–Wunderer [4], where two competing estimates are compared with experiments. We will follow the convention from [4] and call these the 2008-Estimate and the 2016-Estimate. We summarize the two estimates in the following proposition.

Proposition 7 (Heuristic/Experimental UniqueSVP Run-Time). *Let L be a lattice of dimension N and discriminant D. Let $\lambda_1(L)$ be the first minimum of L, i.e., the length of a shortest non-zero vector, and let δ be the Hermite root factor achievable by a lattice reduction algorithm, e.g., as given in Proposition 3 for BKZ-β. Then the algorithm will solve the* SVP[17] *if*:[18]

(a) (2008-Estimate [18, Section 2.2.1])

$$\lambda_1(L) \leq \tau^{-1}\sqrt{N/2\pi e} \cdot \delta^{-N} \cdot D^{1/N},$$

where τ is an experimentally derived constant that appears to be in the range $0.3 \leq \tau \leq 0.5$ for lattices of the form (3).[19]

(b) (2016-Estimate [4, 6])

$$\lambda_1(L) \leq \sqrt{N/\beta} \cdot \delta^{-(N-2\beta)} \cdot D^{1/N},$$

Remark 11. In [4] the authors conclude that the 2016-Estimate predicts that UniqueSVP is easier to solve than predicted by the 2008-Estimate, a prediction that they confirm by experiments on lattices using typical LWE parameters. As we will see in Section 10.4, the opposite is true for the $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT parameters in Tables 19 and 20, i.e., the 2008-Estimate says that the problem is harder than predicted by the 2016-Estimate. The primary reason for the discrepency between out results and those in [4] appears to be the high dimensions used in our parameter sets. In any case, we always insist that our parameters be secure under both the 2008 and the 2016 estimates.

10 Analysis of Lattice Security for $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT

In this section we analyze various ways to use lattice reduction in order to find a key or forge a signature for $\mathsf{PASS}_{\mathtt{RS}}$ or MMSAT.

[17] or at least, the algorithm will able to distinguish whether the lattice contains a non-zero vector that is significantly shorter than expected by the Gaussian heuristic

[18] The upper bounds in (a) and (b) use Stirling's formula to estimate the Γ function.

[19] Although experiments done for BLISS [15] and $\mathsf{PASS}_{\mathtt{RS}}$ [21] gave $\tau \approx 0.48$.

10.1 The Fake Key Hermite Root Factor for $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT

In this section we analyze the security of $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT against a lattice attack designed to find a fake signing key. The analysis of MMSAT key security uses a slightly fancier version of the central limit theorem, which we state here, as well as an elementary calculus calculation about points on spheres.

Theorem 4 (Generalized (Lyapunov) Central Limit Theorem). *Suppose that $\mathcal{X}_1, \mathcal{X}_2, \ldots$ are independent random variables with finite means $\mu_1, \mu_2, \ldots$ and finite variances $\sigma_1^2, \sigma_2^2, \ldots$. For $K \geq 1$, let*

$$s_K^2 = \sigma_1^2 + \cdots + \sigma_K^2 \quad \textit{and} \quad \mathcal{Z}_K = (\mathcal{X}_1 - \mu_1) + \cdots + (\mathcal{X}_K - \mu_K).$$

Then under a technical condition that is satisfied in our applications,[20]

$$\lim_{K\to\infty} \mathrm{Prob}\Big(|\mathcal{Z}_K| \leq C s_K\Big) = \mathsf{erf}\left(C/\sqrt{2}\right).$$

Proof. See for example [38, Theorem 1.15 and equation (197)].

Lemma 1. *Let $S^{N-1} \subset \mathbb{R}^N$ denote the unit sphere taken with the uniform probability measure, and let X_i denote any one of the coordinate functions on $\mathbb{R}^N$. Then the mean of X_i is 0, and the variance of X_i is $1/N$.*

Proof. More generally, it is an elementary calculus exercise to compute all of the moments of X_i. For the convenience of the reader, we include the proof in Section B.

Proposition 8 (Key Security HRF). *Let $\widehat{\boldsymbol{f}}|_\Omega$ be a given $\mathsf{PASS}_{\mathtt{RS}}$ public key. We give Hermite root bounds $\delta_{\text{fake-}\boldsymbol{F}}$ with the following property: A lattice reduction algorithm must achieve an HRF smaller than $\delta_{\text{fake-}\boldsymbol{F}}$ on the $\mathsf{PASS}_{\mathtt{RS}}$ lattice of dimension N and discriminant q^t associated to $\widehat{\boldsymbol{f}}|_\Omega$ in order to find a fake key $\boldsymbol{F}$ that can be used to* (a) *forge a $\mathsf{PASS}_{\mathtt{RS}}$ signature, or to* (b) *add a forged $\mathsf{PASS}_{\mathtt{RS}}$ signature to an existing* MMSAT *aggregate signature.*

(a) *The* fake key HRF *for the $\mathsf{PASS}_{\mathtt{RS}}$ parameter set $(\lambda, N, q, t, d_c, k, b_k)$ is the quantity*[21]

$$\delta_{\text{fake-}\boldsymbol{F}} = \left[\frac{(k-b)\sqrt{N}}{2\sqrt{d_c} q^{t/N}\, \mathsf{erf}^{-1}(2^{-\lambda/N})}\right]^{1/(N-1)}.$$

(b) *The* fake key HRF *for the* MMSAT *parameter set $(\lambda, K, N, q, t, d_c, k, b, B_k)$ is the quantity*

$$\delta_{\text{fake-}\boldsymbol{F}} = \left[\frac{N}{2d_c q^{2t/N}}\left(\frac{B_k^2 K (k-b)^2}{2\,\mathsf{erf}^{-1}(2^{-\lambda/N})^2} - \frac{1}{3}(K-1)(k-b)^2\right)\right]^{1/(2N-2)}.$$

(*We note that the estimate in* (a) *is simply the estimate in* (b) *with $K = B_k = 1$, i.e., forging a $\mathsf{PASS}_{\mathtt{RS}}$ signature is essentially the same as appending a forged $\mathsf{PASS}_{\mathtt{RS}}$ signature to an empty aggregate signature.*)

[20] The required condition is that there is a $\epsilon > 0$ so that $s_K^{-2-\epsilon} \sum_{i=1}^K E\big[|\mathcal{X}_i - \mu_i|^{2+\epsilon}\big] \to 0$ as $K \to \infty$.

[21] Note that erf^{-1} is the inverse of error function, not the reciprocal.

Proof. (a) Let $\boldsymbol{f} \in \mathcal{T}_N(f_f)$ be a $\mathsf{PASS}_{\mathtt{RS}}$ signing key, with corresponding verification key $\widehat{\boldsymbol{f}}|_\Omega$. A polynomial $\boldsymbol{F} \in R_q$ can be used to forge an individual $\mathsf{PASS}_{\mathtt{RS}}$ signature if $\widehat{\boldsymbol{F}}|_\Omega = \widehat{\boldsymbol{f}}|_\Omega$, and further if for every message digest μ, it is feasible to find a $\boldsymbol{y} \in R_q$ so that

$$\|\boldsymbol{F} \star \boldsymbol{c} + \boldsymbol{y}\|_\infty \le k - b \quad \text{with} \quad \boldsymbol{c} = \mathsf{Hash}_{\mathsf{C}}(\widehat{\boldsymbol{y}}|_\Omega, \widehat{\boldsymbol{F}}|_\Omega, \mu) \tag{23}$$

Note that there is no requirement that $\boldsymbol{F}$ have coefficients in $\{-1, 0, 1\}$, nor is there any condition imposed on the coefficients of $\boldsymbol{y}$.[22] Thus for example, the forger could choose the coefficients of $\boldsymbol{y}$ from $\{-1, 0, 1\}$, i.e., $\boldsymbol{y} \in \mathcal{B}^\infty(1)$, in which case the norm bound (23) is essentially a norm bound on $\boldsymbol{F} \star \boldsymbol{c}$. So we will say that the forgery succeeds if $\|\boldsymbol{F} \star \boldsymbol{c}\|_\infty$ is small enough with probability at least $2^{-\lambda}$, i.e., if

$$\operatorname*{Prob}_{\boldsymbol{c} \leftarrow \mathcal{T}_N(d)} \Big(\|\boldsymbol{F} \star \boldsymbol{c}\|_\infty \le k - b \Big) \ge 2^{-\lambda}.$$

Each coefficient of $\boldsymbol{F} \star \boldsymbol{c}$ is a sum of $2d_c$ coefficients of $\pm\boldsymbol{F}$, so if we view $\boldsymbol{F}$ as a random point on a sphere of radius $\|\boldsymbol{F}\|_2$, then each coefficient of $\boldsymbol{F} \star \boldsymbol{c}$ is roughly a sum of $2d_c$ iid random variables chosen from the coordinates of points on that sphere.[23]

Thus the distribution of the sum may be approximated by a normally distributed random variable of mean 0 and variance

$$\|\boldsymbol{F}\|_2^2 / N, \tag{24}$$

where the variance follows from Lemma 1. Treating the coefficients of $\boldsymbol{F} \star \boldsymbol{c}$ as independent, the central limit theorem gives the approximation

$$\begin{aligned} \operatorname*{Prob}_{\boldsymbol{c} \leftarrow \mathcal{T}_N(d_c)} \Big(\|\boldsymbol{F} \star \boldsymbol{c}\|_\infty \le C \Big) &\approx \operatorname*{Prob}_{\boldsymbol{c} \leftarrow \mathcal{T}_N(d_c)} \Big(\big|(\boldsymbol{F} \star \boldsymbol{c})_0\big| \le C \Big)^N && \text{independence assumption,} \\ &\approx \mathsf{erf}\left(\frac{C\sqrt{N}}{2\sqrt{d_c}\,\|\boldsymbol{F}\|_2} \right)^N && \text{Theorem 1 with } K = 2d_c \\ & && \text{and } \sigma = \|\boldsymbol{F}\|_2 / \sqrt{N}. \end{aligned}$$

Taking $C = k - b$, which is the L_∞-norm bound for a valid signature, we see that the forgery is successufl if and only if the forger has found a vector $\boldsymbol{F}$ satisfying

$$\mathsf{erf}\left(\frac{(k-b)\sqrt{N}}{2\sqrt{d_c}\,\|\boldsymbol{F}\|_2} \right)^N \ge 2^{-\lambda}. \tag{25}$$

[22] Although taking $\boldsymbol{y} = \boldsymbol{0}$ would not be a good idea, since then $\widehat{\boldsymbol{y}}|_\Omega = \boldsymbol{0}$, which would suggest that the signature is forged.

[23] It would be more accurate to say that the $\mathcal{Y}_i$ are determined by first choosing a random point on $S^{N-1}(R_\delta)$ and then choosing distinct random coordinates of that point. However, the coordinates of the points of $S^{N-1}(R_\delta)$ are only slightly correlated, so since in practice d_c is quite small compared to N, it suffices to take the $\mathcal{Y}_i$ to be random coordinates of independently chosen random points of $S^{N-1}(R_\delta)$.

By definition of HRF, the smallest vector $\boldsymbol{F}$ obtained by a lattice reduction algorithm (on a $\mathsf{PASS}_{\mathtt{RS}}$ lattice of dimension N and discriminant q^t) has length

$$\|\boldsymbol{F}\|_2 = \mathrm{HRF}^{N-1} \cdot q^{t/N}. \tag{26}$$

Substituting (26) into (25) and solving for the HRF gives the stated result.
(b) Let $\widehat{\boldsymbol{f}}|_\Omega$ be a public key. If the forger is able to achieve an HRF of δ on the associated $\mathsf{PASS}_{\mathtt{RS}}$ lattice of of dimension N and discriminant q^t, then she can find a lattice vector $\boldsymbol{F}$ satisfying

$$\widehat{\boldsymbol{F}}|_\Omega = \widehat{\boldsymbol{f}}|_\Omega \quad \text{and} \quad \|\boldsymbol{F}\|_2 \approx R_\delta = \delta^{N-1} \cdot q^{t/N}. \tag{27}$$

Next suppose that the forger wants to append a forged signature to $\boldsymbol{z}$, which is an existing aggregate of $K-1$ signatures. To do this, she chooses an arbitrary $\boldsymbol{y}$ of her choice, she hashes $\boldsymbol{y}$ $\widehat{\boldsymbol{F}}|_\Omega$, and a document digest μ to create $\boldsymbol{c}$, and then she hopes that

$$\boldsymbol{z}' = \boldsymbol{z} + \boldsymbol{F} \star \boldsymbol{c} + \boldsymbol{y}$$

is a valid aggregate signature. The partial Fourier transform for $\boldsymbol{z}'$ works, so the forger just needs $\|\boldsymbol{z}'\|_\infty$ to be sufficiently small. It is in the forger's best interest to choose $\boldsymbol{y}$ to be small, say $\boldsymbol{y} \in \mathcal{B}^\infty(1)$, so we ignore $\boldsymbol{y}$ and declare the forgery a success if

$$\boldsymbol{z}' = \boldsymbol{z} + \boldsymbol{F} \star \boldsymbol{c} \quad \text{satisfies} \quad \|\boldsymbol{z}'\|_\infty \le B_k\sqrt{K}(k-b).$$

As usual, we treat the coefficients of $\boldsymbol{z}'$ as being essentially independent, so

$$\mathrm{Prob}\Big(\|\boldsymbol{z}'\|_\infty \le B_k\sqrt{K}(k-b)\Big) = \mathrm{Prob}\Big(\big|(\boldsymbol{z}')_0\big| \le B_k\sqrt{K}(k-b)\Big)^N.$$

The 0th coefficient of $\boldsymbol{z}'$ has the form

$$(\boldsymbol{z}')_0 = \sum_{i=1}^{K-1} \beta_i(\boldsymbol{z}_i)_0 + \sum_{i=0}^{N-1} F_{N-i}c_i.$$

The β_i are ±1, while the c_i consist of $2d_c$ copies of ±1 and $N-2d_c$ copies of 0. The coefficients of F_i are the coefficients of a random point on the sphere of radius R_δ. Hence we can model the distribution of $(\boldsymbol{z}')_0$ by the random variable

$$\mathcal{Z} = \sum_{i=1}^{K-1} \mathcal{X}_i + \sum_{i=1}^{2d_c} \mathcal{Y}_i,$$

where the $\mathcal{X}_i$ and $\mathcal{Y}_i$ are independent random variables satisfying

$$\begin{aligned}
\mathcal{X}_i &= \text{random variable uniformly distributed on } \big[-(k-b), k-b\big],\\
\mathcal{Y}_i &= \text{first coordinate of point uniformly distributed on } S^{N-1}(R_\delta).
\end{aligned}$$

Then $\mathcal{X}_i$ and $\mathcal{Y}_i$ have mean 0, and their variances are given by

$$\sigma^2(\mathcal{X}_i) = \frac{1}{3}(k-b)^2 \quad \text{and} \quad \sigma^2(\mathcal{Y}_i) = \frac{1}{N}R_\delta^2,$$

where the former is an easy computation and the latter follows from Lemma 1.

We are going to use the generalized central limit theorem (Theorem 4), which says that if we set

$$s^2 = \sum_{i=1}^{K-1} \sigma^2(\mathcal{X}_i) + \sum_{i=1}^{2d_c} \sigma^2(\mathcal{Y}_i) = \frac{K-1}{3}(k-b)^2 + \frac{2d_c}{N} R_\delta^2, \tag{28}$$

then $\mathcal{Z}/s$ is approximately a standard normal distribution with mean 0 and variance 1. Hence for all C,

$$\text{Prob}\Big(|\mathcal{Z}| \le Cs\Big) \approx \mathsf{erf}\left(C/\sqrt{2}\right).$$

Putting all of this together, we find that the probability of a successful forgery is approximately

$$\begin{aligned}\text{Prob}\Big(\|\boldsymbol{z}'\|_\infty \le B_k\sqrt{K}(k-b)\Big)& \\ \approx \mathsf{erf}\left[\frac{B_k\sqrt{K}(k-b)}{\sqrt{2}\cdot\sqrt{((K-1)/3)(k-b)^2 + (2d_c/N)R_\delta^2}}\right]^N&. \end{aligned}\tag{29}$$

We substitute $R_\delta = \delta^{N-1}\cdot q^{t/N}$ from (27), set (29) equal to $2^{-\lambda}$, and solve for δ to find the HRF for λ-bit lattice key security for MMSAT.

Remark 12. The tail of the error function erf is rapidly decreasing, but on the other hand, the value of erf in the right-hand side of (29) is raised to a large power. This means that as R_δ decreases, there will be a fairly abrupt boundary at which forgery goes from being infeasible to being easy. This occurs roughly when the two terms in the sum (28) defining s^2 are equal.

Example 2. We illustrate Remark (12) with a numerical example. We consider the MMSAT parameter set

$$(\lambda, K, N, q, t, d_c, k, b, B_k) = (256, 1000, 7393, 2^{26.8}, 2464, 29, 309338, 58, 3)$$

from Table 19. The formula in Proposition 8(b) says that this gives the key security HRF

$$\delta_{\text{fake-}\boldsymbol{F}} = 1.00175.$$

According to (13) and (14), at security level $\lambda = 256$, an attacker can run BKZ-β with blocksize $\beta = 967$ and obtain an output whose HRF is $\delta_{\text{BKZ}} = 1.00210$, so forgery is not possible.

We might ask what blocksize is required to achieve an HRF of 1.00175. The output from BKZ-β with blocksize β gives a vector $\boldsymbol{F}$ satisfying

$$R_\delta = \left(\sqrt{\frac{\beta}{2\pi e}}\cdot(\beta\pi)^{1/2\beta}\right)^{(N-1)/(\beta-1)} \cdot q^{t/N},$$

and this value may then be used in the success probability formula (29),

$$\text{Prob(success)} \approx \mathsf{erf}\left[\frac{B_k\sqrt{K}(k-b)}{\sqrt{2}\cdot\sqrt{(K/3)(k-b)^2+(2d_c/N)R_\delta^2}}\right]^N.$$

Table 13 illustrates the abrupt shift from infeasible forgery to easy forgery as the blocksize increases.

Blocksize	Prob(success)	BKZ run-time
967	$2^{-24283.73}$	$2^{256.27}$
1242	$2^{-827.25}$	$2^{329.13}$
1261	$2^{-464.62}$	$2^{334.17}$
1280	$2^{-240.38}$	$2^{339.20}$
1299	$2^{-114.64}$	$2^{344.24}$
1317	$2^{-53.11}$	$2^{349.01}$
1336	$2^{-22.27}$	$2^{354.04}$
1355	$2^{-9.00}$	$2^{359.08}$

Table 13: Example of Probability of Forgery Success with Increasing Blocksize

10.2 Solving for the $\mathsf{PASS}_{\mathsf{RS}}$ Private Key as a UniqueSVP

In this section we analyze the difficulty of finding a $\mathsf{PASS}_{\mathsf{RS}}$ private key formulated as a UniqueSVP. We note that the private key $\boldsymbol{f}$ is a vector length $\sqrt{2d_f}$ in the $\mathsf{PASS}_{\mathsf{RS}}$ lattice (3) of dimension N and discriminant q^t, so the SVP-Gap is

$$\mathsf{SVP\_Gap} = \frac{\mathsf{GH}}{\lambda_1} \approx \sqrt{\frac{N}{4d_f\pi e}}\cdot q^{t/N},$$

which will be significantly larger than 1. (Note that we always have $d_f < \frac{1}{3}N$.)

We recall from Proposition 7 that there are two competing estimates for the difficulty of solving UniqueSVP. The conclusion of [4] is that the 2016-Estimate predicts that UniqueSVP is easier to solve than predicted by the 2008-Estimate, and that this prediction is confirmed by experiments on lattices using typical LWE parameters. We were thus somewhat surprised that for the parameters used by $\mathsf{PASS}_{\mathsf{RS}}$ and MMSAT, the opposite is true, i.e., the 2016-Estimate says that the UniqueSVP for finding a $\mathsf{PASS}_{\mathsf{RS}}$ secret key is *harder* than predicted by the 2008-Estimate! To see why, we solve the two estimates for the HRF δ required to solve the UniqueSVP:

$$\text{2008-Estimate} \qquad \delta = \left[\frac{1}{\tau\sqrt{2\pi e}}\cdot\sqrt{N}\cdot D^{1/N}\cdot\lambda_1(L)^{-1}\right]^{1/N}, \tag{30}$$

$$\text{2016-Estimate} \qquad \delta = \left[\frac{1}{\sqrt{\beta}} \cdot \sqrt{N} \cdot D^{1/N} \cdot \lambda_1(L)^{-1}\right]^{1/(N-2\beta)}. \tag{31}$$

The estimates differ in two ways. First, the $1/\tau\sqrt{2\pi e} \approx 0.6$ factor in the 2008-Estimate is replaced by $1/\sqrt{\beta}$ in the 2016-Estimate. For security levels 128 and 256, as given for example in Table 19, the allowed blocksizes are, respectively, $\beta = 484$ and $\beta = 966$, so $1/\sqrt{\beta}$ is 0.046 and 0.032. This means that the quantity in square brackets is roughly 13 to 18 times smaller for the 2016-Estimate.

Second, the 2008-Estimate is raised to the $1/N$ power, while the 2016-Estimate is raised to the smaller $1/(N-2\beta)$ power. If β is large, this will definitely tend to make the 2016-Estimate larger than the 2008-Estimate, but there is a delicate balance of whether it compensates for the fact that the 2008-Estimate started 13 to 18 times smaller before taking the roots. Table 14 shows what happens for some typical parameter sets from Table 19.

λ	128	256
β	484	967
N	4111	7393
q	$2^{24.909}$	$2^{26.806}$
t	1307	2464
$D^{1/N} = q^{t/N}$	315.4	489.1
d_f	1307	2464
$\lambda_1 = \sqrt{2d_f}$	52.35	70.20
$\mathsf{GH} \approx \sqrt{N/2\pi e} \cdot q^{t/N}$	4893	10177
Estimate-2008 δ	1.0013	1.00080
Estimate-2016 δ	1.0009	1.00054
δ for BKZ-β	1.0035	1.00210

Table 14: Comparision of the UniqueSVP HRF for the 2008-Estimate versus the 2016-Estimate

10.3 HRF Required to Forge the z Part of an MMSAT Signature

We use the result in Proposition 5 to estimate the HRF needed to forge an MMSAT signature $\boldsymbol{z}$. We note that this estimate includes the L_∞-norm versus L_2-norm correction discussed in Section 9.3.

Proposition 9. *For the* $\mathsf{PASS}_{\mathtt{RS}}$ *lattice* (3) *of dimension* N *and discriminant* q^t*, in order to find a* $\boldsymbol{z}$ *that forges an* MMSAT *signature, the forger needs to achieve an HRF smaller than*

$$\delta_{L_\infty\text{-forge}} = \left(\frac{2\Gamma(1+N/2)^{1/N} B_q q^{1-t/N}}{\sqrt{\pi}}\right)^{1/(N-1)}. \tag{32}$$

Proof. For the MMSAT aggregate signatures, the L_∞-norm bound in the $\mathsf{PASS}_{\mathtt{RS}}$ lattice is $C = B_q q$, and the discriminant of the lattice is $\mathrm{Disc} = q^t$. Substituting these values into Proposition 5 gives the stated result.

Remark 13. For $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT, it is not literally true that the attacker succeeds if they find a vector satisfying (32), since the given basis for the lattice includes the q-vectors $q\boldsymbol{e}_1, \ldots, q\boldsymbol{e}_t$. Thus the lattice contains t mutually orthogonal vectors whose L_2-norms are equal to q, so it contains lots of vectors satisfying (32), specifically all linear combinations

$$n_1 q\boldsymbol{e}_1 + \cdots + n_t q\boldsymbol{e}_t \quad \text{with} \quad n_1^2 + \cdots + n_t^2 \le 2NB_q^2/\pi e.$$

Of course, these vectors are useless for forging, since their L_∞-norms are larger than $B_q q$. So when a forger uses BKZ to try to find a vector satisfying (32) or some similar inequality, at the very least they are actually looking for a vector that is not built solely out of the known q-vectors.

10.4 A Summary of the Hermite Root Factors for MMSAT Lattice Security

For a given desired bit security level λ, we use Proposition 4 to set the achievable block-size $\beta = \lceil \lambda/0.265 \rceil$, and then Proposition 3 tells us the HRF that is achievable by BKZ-β.

We have described a number of ways in which a lattice reduction algorithm such as BKZ-β, applied to the lattice (3), can be used to attack $\mathsf{PASS}_{\mathtt{RS}}$ or MMSAT. These include forging the signature $\boldsymbol{z}$, finding a vector $\boldsymbol{F}$ that can be used in place of the private key $\boldsymbol{f}$, or directly searching for $\boldsymbol{f}$. Each of these lattice problems has an associated HRF, and the security of $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT requires that the HRFs for the attacks be smaller than the HRF achievable by BKZ β. We summarize these requirements in the next proposition.

Proposition 10 (Lattice Security of $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT). *Let*

$$(\lambda, K, N, q, t, d_c, k, b, B_k)$$

be an MMSAT *parameter set, or alternatively a* $\mathsf{PASS}_{\mathtt{RS}}$ *parameter set in which* $K = B_k = 1$. *Set* $\beta = \lceil \lambda/0.265 \rceil$, *and define*[24]

$$\begin{aligned} \delta_\beta &= \text{the HRF achieved by BKZ-}\beta \text{ for blocksize } \beta = \lambda/0.265 \\ &= \left[\frac{\beta}{2\pi e} \cdot (\beta\pi)^{1/\beta} \right]^{1/(2\beta-2)}, \end{aligned}$$

$$\begin{aligned} \delta_{\text{forge-}\boldsymbol{z}} &= \text{the HRF required to forge an amalgamate signature } \boldsymbol{z} \\ &= \left[\frac{2\Gamma(1+N/2)^{1/N} B_q q^{1-t/N}}{\sqrt{\pi}} \right]^{1/(N-1)}, \end{aligned}$$

[24] Here τ is an experimentally determined constant. A conservative choice is $\tau = 1/3$.

$$\delta_{\text{fake-}\boldsymbol{F}} = \text{the HRF required to find an } \boldsymbol{F} \text{ that serves a a fake signing key}$$
$$= \left[\frac{N}{2d_c q^{2t/N}}\left(\frac{B_k^2 K(k-b)^2}{2\,\mathsf{erf}^{-1}(2^{-\lambda/N})^2} - \frac{1}{3}(K-1)(k-b)^2\right)\right]^{1/(2N-2)},$$

$$\delta_{\text{USVP-08}} = \text{the HRF required to solve the } \mathsf{UniqueSVP} \text{ and find } \boldsymbol{f} \text{ (2008-Estimate)}$$
$$= \left[\frac{1}{\tau\sqrt{4d_f\pi e}} \cdot \sqrt{N} \cdot q^{t/N}\right]^{1/N},$$

$$\delta_{\text{USVP-16}} = \text{the HRF required to solve the } \mathsf{UniqueSVP} \text{ and find } \boldsymbol{f} \text{ (2016-Estimate)}$$
$$= \left[\frac{1}{\sqrt{2d_f\beta}} \cdot \sqrt{N} \cdot q^{t/N}\right]^{1/(N-2\beta)}.$$

Then the given MMSAT (*or* $\mathsf{PASS}_{\mathtt{RS}}$) *parameter set is secure against* (known) *lattice attacks provided that*

$$\delta_\beta > \max\{\delta_{\text{forge-}\boldsymbol{z}}, \delta_{\text{fake-}\boldsymbol{F}}, \delta_{\text{USVP-08}}, \delta_{\text{USVP-16}}\}.$$

Proof. For the derivation of these HRF formulas, see Proposition 3 for δ_β, see Proposition 9 for $\delta_{\text{forge-}\boldsymbol{z}}$, see Proposition 8(b) for $\delta_{\text{fake-}\boldsymbol{F}}$, see Proposition 7(a) and (30) for $\delta_{\text{USVP-08}}$, and see Proposition 7(b) and (31) for $\delta_{\text{USVP-16}}$.

11 Choosing Parameters for MMSA and MMSAT

Proposition 4 says that a conservative run-time estimate for BKZ-β is[25]

$$\log_2(\text{BKZ-}\beta \text{ Run-Time}) \approx 0.265 \cdot \beta. \tag{33}$$

And as noted in Proposition 3, the following estimate for the HRF for the output of BKZ-β seems to be consistant with experiments assuming reasonably large blocksize ($\beta \geq 50$) and lattice dimension significantly larger than the blocksize:

$$\delta(\text{BKZ-}\beta) \approx \left[\frac{\beta}{2\pi e} \cdot (\pi\beta)^{1/\beta}\right]^{1/(2\beta-2)}. \tag{34}$$

Using (33) and (34) as our base formulas, we now develop an algorithm to choose $\mathsf{PASS}_{\mathtt{RS}}$ and MMSAT parameters.

1. Select the following quantities:

   $\lambda \leftarrow$ bit security level, typically in $[128, 4096]$.

   $K \leftarrow$ maximum number of signatures to be aggregated, typically in $[10^3, 10^6]$.

[25] Indeed, it is not clear that anyone will achieve even 0.29β in the foreseeable future, much less 0.265β, without some major theoretical breakthrough; cf. Section 9.1.

$B_k \leftarrow$ an aggregate signature verification scaling factor, typically in $[2, 6]$.
$B_q \leftarrow$ an aggregate signature verification scaling factor, typically in $\left[\frac{1}{4}, \frac{1}{3}\right]$.
$B_t \leftarrow$ lattice scaling factor, typically in $\left[\frac{1}{3}, \frac{1}{2}\right]$.
$p \leftarrow$ desired single signature accept rate, typically between 10% and 99%.
(For $\mathsf{PASS}_{\mathtt{RS}}$, set $K = B_k = 1$.)

2. Output consists of the following quantities:

$N =$ dimension, a prime
$q =$ modulus, a prime satisfying $q \equiv 1 \pmod{N}$
$t =$ number of coordinates in partial Fourier transform, $t \approx B_t N$
$k =$ norm bound for challenge polynomial
$b =$ norm bound displacement for rejection sampling of signatures
$d_c =$ number of 1 and -1 coefficients in commitment $\boldsymbol{c}$
$d_f =$ number of 1 and -1 coefficients in private key $\boldsymbol{f}$

3. Use (33) to compute the allowable blocksize for the given bit security level:

$$\beta \leftarrow \lambda / 0.265.$$

4. Use (34) to compute the BKZ-β HRF for the given bit security level,

$$\delta_\beta \leftarrow \left[\frac{\beta}{2\pi e} \cdot (\pi\beta)^{1/\beta}\right]^{1/(2\beta-2)}.$$

5. The goal now is to find parameters satisfying the following inequalities:

$$B_k\sqrt{K}(k-b) \approx B_q q \qquad \text{signature norm bound} \qquad (35)$$
$$b = 2d_c \qquad \text{rejection sampling offset} \qquad (36)$$
$$\binom{N}{d_c, d_c} \geq 2^{2\lambda} \qquad \text{combinatorial security} \qquad (37)$$
$$\left(\sqrt{N} B_q q^{1-B_t}\right)^{1/N} \lessapprox \delta_\beta \qquad \text{lattice forgery security} \qquad (38)$$
$$\left(1 - \frac{b}{k}\right)^N \gtrapprox p \qquad \text{acceptance probability} \qquad (39)$$

6. Substituting (36) into (39) gives

$$k \geq \frac{2d_c}{1 - p^{1/N}}. \qquad (40)$$

7. Using (36) and (40) in (35), a bit of algebra gives a lower bound for q, while (38) gives an upper bound for q. Thus

$$\frac{2B_k\sqrt{K}}{B_q} \cdot d_c \cdot \frac{p^{1/N}}{1 - p^{1/N}} \leq q \leq \left(\frac{\delta_\beta^N}{B_q\sqrt{N}}\right)^{1/(1-B_t)}. \qquad (41)$$

We note that $1 - p^{1/N} \sim N/\log(p^{-1})$ as $N \to \infty$, so the lower bound in (41) grows linearly with N. On the other side, the upper bound grows exponentially, since $\delta_\beta > 1$. Hence if N is sufficiently large, there will be a non-empty interval allowed for q.

8. All of the quantities in (41) except for N, q, and d_c are preset parameters, and d_c can only decrease as N increases. The goal now is to find a triple (N, q, d_c) satisfying (37) and (41). And since the public key size is roughly $B_t N \log_2(q)$, we would like to find the triple that minimizes $N \log_2(q)$.
9. This leads to the following algorithm, which is briefly described here, and which is given in full detail as a working algorithm in Table 15.
   (a) Find the smallest N so that the interval (41) with $d_c = 1$ is non-empty.
   (b) Set d_c to be the smallest value so that $\binom{N}{d_c,d_c} \geq 2^{2\lambda}$. This ensures that the inequality (37) holds.
   (c) Set N to be the smallest prime so that the interval (41) contains a prime q satisfying $q \equiv 1 \pmod{N}$, and set q equal to that value.
   (d) Assign the following values:

$$t \leftarrow \lfloor B_t N \rfloor, \qquad b \leftarrow 2d_c, \qquad k \leftarrow \left\lfloor B_q q / B_k \sqrt{K} + b \right\rceil.$$

   (e) Typically one sets $d_f \leftarrow \lfloor N/3 \rfloor$, which maximizes key security. Alternatively, one may take a smaller d_f, provided that $\binom{N}{d_f,d_f} \geq 2^{2\lambda}$.
   (f) If desired, recompute $p \leftarrow (1 - b/k)^N$ to see how much its value has changed due to rounding.
   (g) Compute the quantities $\delta_{\text{forge-}\boldsymbol{z}}$, $\delta_{\text{fake-}\boldsymbol{F}}$, $\delta_{\text{USVP-08}}$, $\delta_{\text{USVP-16}}$ using the formulas in Proposition 10. If

$$\delta_\beta > \max\{\delta_{\text{forge-}\boldsymbol{z}}, \delta_{\text{fake-}\boldsymbol{F}}, \delta_{\text{USVP-08}}, \delta_{\text{USVP-16}}\} \tag{42}$$

   output a successful parameter set. Otherwise select a larger prime N and try again. (N.B. Our algorithm is designed to create parameters satisfying $\delta_\beta > \delta_{\text{forge-}\boldsymbol{z}}$, since in practice we have found that $\delta_{\text{forge-}\boldsymbol{z}}$ tends to be the largest of the four δs on the right-hand side of (42).)

Remark 14. We note again that the forgery bound is an L_∞-norm bound, not an L_2-norm bound. Thus as explained in Remark 8, the $\sqrt{N}$ in the lattice forgery inequality (38) can be strengthened by replacing it with $2\Gamma(1 + N/2)^{1/N}/\sqrt{\pi} \approx \sqrt{2N/\pi e}$ inside the parenthesis. Hence, in practice we replace (38) with the forgery requirement

$$\left(\frac{2\Gamma(1 + N/2)^{1/N} B_q q^{1-t/N}}{\sqrt{\pi}} \right)^{1/(N-1)} \leq \delta_\beta. \tag{43}$$

Similar considerations apply to the key recovery problem.

Algorithm 9 Choosing MMSAT Parameters

Input:
- λ bit security level, typically in $[128, 4096]$.
- K maximum number of signatures to be aggregated, typically in $[10^3, 10^6]$.
- B_k aggregate signature verification scaling factor, typically in $[2, 6]$.
- B_q aggregate signature verification scaling factor, typically in $\left[\frac{1}{4}, \frac{1}{3}\right]$.
- B_t lattice scaling factor, typically in $\left[\frac{1}{3}, \frac{1}{2}\right]$.
- p single signature accept rate, typically between 10% and 99%.

Output:
- N dimension, a prime
- q modulus, a prime satisfying $q \equiv 1 \pmod{N}$
- t number of coordinates in partial Fourier transform
- k norm bound for challenge polynomial
- b used for norm bound/rejection sampling of signatures
- d_c numbers of 1 and -1 coefficients in commitment c
- d_f numbers of 1 and -1 coefficients in private key f

Useful Non-Parameter Quantities
- β BKZ blocksize
- δ_β HRF for BKZ β at bit security λ
- $\delta_{\text{forge-}\boldsymbol{z}}$ HRF to forge the $\boldsymbol{z}$ part of an MMSAT signature
- $\delta_{\text{fake-}\boldsymbol{F}}$ HRF to find an $\boldsymbol{F}$ that works as a signing key
- $\delta_{\text{USVP-08}}$ HRF to find $\boldsymbol{f}$ as UniqueSVP — 2008 Estimate
- $\delta_{\text{USVP-16}}$ HRF to find $\boldsymbol{f}$ as UniqueSVP — 2016 Estimate

1: $\beta \leftarrow \lambda/0.265$ // Blocksize from (33)

2: $\delta_\beta \leftarrow \left[\frac{\beta}{2\pi e} \cdot (\pi\beta)^{1/\beta}\right]^{1/(2\beta-2)}$ // HRF for BKZ-β from (34)

3: $N \leftarrow$ smallest prime such

$$\frac{2B_k\sqrt{K}}{B_q} \cdot \frac{p^{1/N}}{1-p^{1/N}} < \left(\frac{\delta_\beta^N}{B_q\sqrt{N}}\right)^{1/(1-B_t)}$$ // This makes the interval (41) // with $d_c = 1$ non-empty.

4: $d_c \leftarrow$ smallest value so that $\binom{N}{d_c, d_c} \geq 2^{2\lambda}$

5: **while** the interval (41) contains no primes satisfying $q \equiv 1 \pmod{N}$ **do**

6: $N \leftarrow \mathsf{Next\_Prime}(N)$

7: **end while**

8: $q \leftarrow$ smallest prime satisfying (41) and $q \equiv 1 \pmod{N}$

9: $t \leftarrow \lfloor B_t N \rfloor$

10: $b \leftarrow 2d_c$

11: $k \leftarrow \left\lfloor \frac{B_q \cdot q}{B_k\sqrt{K}} + b \right\rfloor$

12: $d_f \leftarrow \lfloor N/3 \rfloor$ // May be smaller, but must satisfy $\binom{N}{d_f, d_f} > 2^{2\lambda}$

13: $p \leftarrow (1 - b/k)^N$ // Recompute signing probability

14: $\delta_{\text{forge-}\boldsymbol{z}} \leftarrow \left(\frac{2\Gamma(1+N/2)^{1/N} B_q q^{1-t/N}}{\sqrt{\pi}}\right)^{1/(N-1)}$

15: $\delta_{\text{fake-}\boldsymbol{F}} \leftarrow \left[\frac{N}{2d_c q^{2t/N}} \left(\frac{B_k^2 K(k-b)^2}{2\,\mathsf{erf}^{-1}(2^{-\lambda/N})^2} - \frac{1}{3}K(k-b)^2\right)\right]^{1/(2N-2)}$

16: $\delta_{\text{USVP-08}} \leftarrow \left[\frac{1}{\tau\sqrt{4d_f\pi e}} \cdot \sqrt{N} \cdot q^{t/N}\right]^{1/N}$ // Typically $\tau = \frac{1}{3}$

17: $\delta_{\text{USVP-16}} \leftarrow \left[\frac{1}{\sqrt{2d_f\beta}} \cdot \sqrt{N} \cdot q^{t/N}\right]^{1/(N-2\beta)}$

18: **if** $\delta_\beta \leq \max\{\delta_{\text{forge-}\boldsymbol{z}}, \delta_{\text{fake-}\boldsymbol{F}}, \delta_{\text{USVP-08}}, \delta_{\text{USVP-16}}\}$ **then**

19: $N \leftarrow \mathsf{Next\_Prime}(N)$

20: **go to** Step 8

21: **end if**

Table 15: Algorithm for Selecting MMSAT Parameters

12 Sample Parameter Sets for MMSAT

Table 19 on page 60 lists a variety of sample parameters that were selected by running Algorithm 9 (Table 15) using various values of $(\lambda, B_k, B_q, B_t, p)$. We illustrate by discussing the first line of that table.

Example 3. We consider a MMSAT parameter set with

$$K = 1000, \quad \lambda = 128, \quad B_k = 3, \quad B_q = \frac{1}{4}, \quad B_t = \frac{1}{3}, \quad \text{Prob(Accept)} = 25.00\%.$$

Algorithm 9 on this input gives parameters

$$(N, q, t) = (4111, 2^{24.909}, 1370), \quad (d_f, d_c) = (1370, 14), \quad (k, b) = (83047, 28).$$

This gives the desired combinatorial security level and accept probability,

$$\#\mathcal{T}_N(d_c) = \binom{N}{d_c, d_c} \approx 2^{263.3} \quad \text{and} \quad \left(1 - \frac{b}{k}\right)^N \approx 25.15\%.$$

The security level $\lambda = 128$ means that a forger is allowed to run BKZ-β with blocksize $\beta = 128/0.265 \approx 484$, which from (13) means that the forger can achieve an HRF of

$$\delta_{\text{BKZ}}(484) = 1.0034836.$$

According to (32), the forger succeeds if $\delta_{\text{BKZ}}(484)$ is smaller that the following forgery HRF for the given parameters (with $B_q = 1/4$):

$$\delta_{\text{forge}}(4111, 2^{24.909}, 1370) \approx 1.0033051.$$

Thus the forgery is not successful. Indeed, we have been somewhat conservative in estimating lattice security, since δ_{forge} is significantly smaller than δ_{BKZ} with $\beta = 484$. We might ask what blocksize is required to successfully forge. The answer is

$$\delta_{\text{BKZ}}(521) = 1.0033055 \quad \text{and} \quad \delta_{\text{BKZ}}(522) = 1.0033010,$$

so blocksize $\beta = 522$ is required, which corresponds to $\lambda = 138.33$. Thus the given parameter set ostensibly has better than 138-bit lattice security. In view of the uncertainties in estimating lattice security, we view this extra 10-bits of lattice security as a feature, not a bug.

See also Table 1 in Section 1.1 for a comparison of parameter sizes of MMSAT and MMSATK with other post-quantum schemes such as BLISS and $\text{PASS}_{\texttt{Rs}}$, as well as with the non-quantum-secure ECDSA scheme.

13 MMSATK Bit Security and Choosing MMSATK Parameters

The lattice forgery problem underlying MMSA is to find a sufficiently short vector $\boldsymbol{z}$ such that the t coordinates of $\widehat{\boldsymbol{z}}|_\Omega$ take on specified values in $\mathbb{Z}_q$. The lattice has dimension N and discriminant q^t.

The lattice forgery problem for MMSATK is similar, except that the forger now only needs to match t' of the coorindates of $\widehat{\boldsymbol{z}}|_\Omega$ to specified values, where $t' < t$. However, the forger does not know in advance which t' coordinates need to match. Indeed, the t' indices are chosen as a hash of z, thereby making it impossible for the forger to preselect any of the t' indices.

However, the MMSATK forger gains some advantage over the MMSA forger. Specifically, she may choose to solve an easier lattice problem in which she finds a sufficiently short $\boldsymbol{z}$ such that t'' of the coordinates of $\widehat{\boldsymbol{z}}|_\Omega$ are correct. If t'' is fairly large compared to t', then there is a reasonable possibility that she will be lucky and the t' indices chosen by hashing z will be a subset of the good t'' coordinates that she forced to be correct.[26] Hence

$$\text{Expected run time to forge an MMSATK signature } \boldsymbol{z}$$
$$= \begin{pmatrix}\text{Expected run time to find } \boldsymbol{z}\\ \text{satisfying } \|\boldsymbol{z}\|_\infty \le B_q q \text{ in a}\\ \mathsf{PASS}_{\mathtt{RS}} \text{ lattice of dimen-}\\ \text{sion } N \text{ and discriminant } q^{t''}\end{pmatrix} \text{Prob}\begin{pmatrix}\text{Choosing } t' \text{ samples from}\\ \text{a set of size } t \text{ containing}\\ t'' \text{ winning tickets and}\\ \text{getting all winning tickets}\end{pmatrix}^{-1}. \quad (44)$$

The probability in (44) is easy to compute. There are $\binom{t}{t'}$ ways to choose the sample, and of those, exactly $\binom{t''}{t'}$ of them consist entirely of winning tickets, so the probability is

$$\text{Prob}\begin{pmatrix}\text{Choosing } t' \text{ samples from}\\ \text{a set of size } t \text{ containing}\\ t'' \text{ winning tickets and}\\ \text{getting all winning tickets}\end{pmatrix} = \frac{\binom{t''}{t'}}{\binom{t}{t'}}$$

To estimate the first factor in (44), we let $\delta_{\text{BKZ}}(\beta)$ be the expected HRF of BKZ-β as given by the formula (13) in Proposition 3, and we will also use the fact that, according to Proposition 4, the cost of running BKZ-β is approximately $2^{0.265\beta}$. On the other hand, the formula (32) in Remark 8 says that the HRF to find a forgery is

$$\delta_{\text{forge}}(t'') = \left(\frac{2\Gamma(1+N/2)^{1/N} B_q q^{1-t''/N}}{\sqrt{\pi}}\right)^{1/(N-1)}.$$

[26] Even if a few of the t' indices are not in the forger's good set, she might be lucky and have the extra coordinates match by chance. But each such coordinate has only a $1/q$ chance of being correct, and q is quite large, so we will ignore this small error term in our calculation.

Note that this formula uses t'' in place of t. This means that the δ_{forge} for MMSATK is larger than the δ_{forge} for MMSA, making MMSATK somewhat easier to forge than MMSA, as we would expect.

For each t'' let $\beta(t'')$ be the smallest β so that so that

$$\delta_{\text{BKZ}}\big(\beta(t'')\big) \leq \delta_{\text{forge}}(t'').$$

Then the bit security of MMSATK against forgery lattice attacks is obtained by taking the optimal (for the forger) value of t'',

$$\text{MMSATK bit security} = \min_{t' \leq t'' \leq t} \left[0.265\beta(t'') + \log_2 \binom{t}{t'} - \log_2 \binom{t''}{t'} \right].$$

Table 16 gives some examples showing that when $B_t = \frac{1}{3}$, i.e., $t \approx \frac{1}{3}N$, this attack decreases the bit security by only a few bits. However, for $B_t = \frac{1}{2}$, the decrease in security is more substantial, bringing the security level below the desired λ value. In practice, the security level can easily be raised back to the desired level by increasing t' slightly, which in turn will make the MMSATK key a bit larger.

Example 4. We illustrate with the parameter set in the last line of Table 16. The target bit security is $\lambda = 256$, but the MMSATK bit security for forgery is only $\lambda_{\text{forge}} = 248.68$. In order to achieve the desired bit security, we need to increase t' from 78 to 92. With this change, we obtain blocksize $\beta_{\text{forge}} = 369$ and $\lambda_{\text{forge}} = 256.38$. The consequence of the larger t' value is that the key size of MMSATK increases from 323 Bytes to 369 Bytes. Although not insignificant, the compressed key size is still is stil almost 30 times smaller than the key size for MMSA.

						MMSA			MMSATK		
$\approx \lambda$	N	q	t	t'	t''	key	β_{forge}	λ_{forge}	key	β_{forge}	λ_{forge}
128	4111	$2^{24.909}$	1370	40	1140	4266 Bytes	522	138.33	157 Bytes	473	136.12
128	5119	$2^{32.333}$	1706	38	1398	6896 Bytes	512	135.68	186 Bytes	460	132.95
128	3217	$2^{24.655}$	1608	39	998	4956 Bytes	531	140.72	153 Bytes	361	122.92
256	7393	$2^{26.806}$	2464	82	2129	8257 Bytes	1033	273.74	339 Bytes	959	271.74
256	9041	$2^{34.102}$	3013	77	2550	12844 Bytes	1018	269.77	393 Bytes	935	266.57
256	5791	$2^{26.503}$	2895	78	1924	9591 Bytes	1048	277.72	323 Bytes	762	248.68

Table 16: Minimal value of t' satisfying (8) for some parameter sets in Table 19, with a comparison of security levels and key sizes for MMSAT and MMSATK.

A Proof of the Reduction of MMSA to PASS_{RS}

In this section we give the proof of the reduction of MMSA to PASS_{RS}.

Proof (of Theorem 3 *in Section* 8*).* The goal of the Challenger $\mathcal{C}$ is to interact with the Chosen Key Forger $\mathcal{A}$ to forge a Single Signature on μ_K using Alice's key $\widehat{\boldsymbol{f}_A}|_\Omega$. Note that $\mathcal{C}$ does not know the secret key $\boldsymbol{f}_A$ and has no influence over how μ_K is chosen. The Challenger $\mathcal{C}$ simulates the forger $\mathcal{A}$ by presenting a challenge key, returning signature and hash queries. In response the Forger $\mathcal{A}$ returns an aggregate signature forgery. The Challenger $\mathcal{C}$ uses the aggregate forgery to extract a $\mathsf{PASS}_{\mathtt{RS}}$ signature. To handle the hash (random oracle) queries, $\mathcal{C}$ maintains a list H. In MMSA hashes are computed with input $(\widehat{\boldsymbol{y}}|_\Omega, \mu, \widehat{\boldsymbol{f}}|_\Omega)$. Therefore, $\mathcal{C}$ simulates the random oracle H with inputs $(\widehat{\boldsymbol{y}}|_\Omega, \mu, \widehat{\boldsymbol{f}}|_\Omega)$.

The simulation proceeds as follows:

Challenge PK: The Challenger $\mathcal{C}$ gives the aggregate forger $\mathcal{A}$ a challenge public key $\widehat{\boldsymbol{f}_K}|_\Omega = \widehat{\boldsymbol{f}_\mathcal{C}}|_\Omega + \widehat{\boldsymbol{f}_A}|_\Omega$ where $\widehat{\boldsymbol{f}_A}|_\Omega$ is Alice's public key and $\widehat{\boldsymbol{f}_\mathcal{C}}|_\Omega$ is the public key of the Challenger. Hence, $\mathcal{C}$ knows $\boldsymbol{f}_\mathcal{C}$ but not $\boldsymbol{f}_K$.

Signature Queries: $\mathcal{A}$ requests signatures on $\mu_1, \ldots, \mu_{N_S}$ under $\widehat{\boldsymbol{f}_K}|_\Omega$. $\mathcal{C}$ prepares signatures $\boldsymbol{z}_i, \boldsymbol{c}_i$ and returns them to $\mathcal{A}$ by iterating the following steps for $i = 1, \ldots, N_S$:

1. Sample $\boldsymbol{y}_i^\mathcal{C} \in \mathcal{B}^\infty(k)$ and $\boldsymbol{c}_i \in \mathcal{B}^\infty(b)$.
2. Compute $\widehat{\boldsymbol{y}_i}|_\Omega = \widehat{\boldsymbol{y}_i^\mathcal{C}}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega \odot \widehat{\boldsymbol{c}_i}|_\Omega$.
3. If entry with matching $\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_1}|_\Omega$ exists in H, restart from Step 1.
4. Compute $\boldsymbol{z}_i = \boldsymbol{f}_\mathcal{C} \star \boldsymbol{c}_i + \boldsymbol{y}_i^\mathcal{C}$.
5. If $\|\boldsymbol{z}_i\| > k - b$ then return to Step 1 (rejection sampling).
6. Uniformly sample $\beta_i \in \{-1, 1\}$.
7. Insert $\langle \widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_K}|_\Omega, \boldsymbol{c}_i, \boldsymbol{z}_i, \beta_i \rangle$ into H.
8. Return signature $(\boldsymbol{z}_i, \boldsymbol{c}_i, \mu_i)$ to $\mathcal{A}$.

Note that Steps 1 and 5 ensure that the signatures are indistinguishable from authentic signatures to the forger $\mathcal{A}$.

Message Hash Queries: The forger $\mathcal{A}$ requests $\mathsf{H}(\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$ for forged keys $\{\widehat{\boldsymbol{f}_i}|_\Omega\}_{i=1}^{N_H}$. The challenger $\mathcal{C}$ handles the hash queries as follows for $i \neq N_H$:

1. If entry for $\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega$ exists in H, lookup and return $\boldsymbol{c}_i$.
2. Otherwise $\mathcal{C}$ samples $\boldsymbol{c}_i \in \mathcal{B}^\infty(b)$ and $\beta_i \in \{-1, +1\}$.
3. Return $\boldsymbol{c}_i$ and insert $\langle \widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega, \boldsymbol{c}_i, \mathtt{null}, \beta_i \rangle$ into random oracle H.

In the last iteration $\mathcal{A}$ queries for $\mathsf{H}(\widehat{\boldsymbol{y}_{N_H}}|_\Omega, \mu_{N_H}, \widehat{\boldsymbol{f}_{N_H}}|_\Omega)$. The Challenger follows these steps instead:

1. If $\widehat{\boldsymbol{f}_{N_H}}|_\Omega \neq \widehat{\boldsymbol{f}_K}|_\Omega$ then $\mathtt{ABORT}$.
2. If $\mu_K, \widehat{\boldsymbol{f}_K}|_\Omega$ is found in H and corresponding $\boldsymbol{z}_i$ is not $\mathtt{null}$ then $\mathtt{ABORT}$.
3. Challenger $\mathcal{C}$ retrieves $\widehat{\boldsymbol{f}_i}|_\Omega, \widehat{\boldsymbol{c}_i}|_\Omega, \widehat{\boldsymbol{y}_i}|_\Omega$ from H for $i = 1, \ldots, N_H - 1$.
4. Challenger $\mathcal{C}$ uniformly samples $\beta_i \in \{-1, 1\}$ and inserts this value into $\mathsf{H}(\widehat{\boldsymbol{f}_i}|_\Omega, \widehat{\boldsymbol{c}_i}|_\Omega, \widehat{\boldsymbol{y}_i}|_\Omega)$.
5. Challenger $\mathcal{C}$ computes $\widehat{\boldsymbol{y}_A}|_\Omega$:

$$\widehat{\boldsymbol{y}_A}|_\Omega = \widehat{\boldsymbol{y}_K}|_\Omega + \beta_K \sum_{i=1}^{N_H - 1} \beta_i (\widehat{\boldsymbol{f}_i}|_\Omega \odot \widehat{\boldsymbol{c}_i}|_\Omega + \widehat{\boldsymbol{y}_i}|_\Omega).$$

6. Challenger $\mathcal{C}$ computes and returns $\boldsymbol{c}_A = \mathsf{Hash}(\widehat{\boldsymbol{y}_A}|_\Omega, \mu_K, \widehat{\boldsymbol{f}_A}|_\Omega)$.

Aggregation Hash Query: For aggregation $\mathcal{A}$ needs the β_i values, and therefore needs make another random oracle query to H with $\{\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega\}_{i=1}^K$ as input. For $(\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_i}|_\Omega)$ entries that were requested earlier and stored in the hash table, the Challenger $\mathcal{C}$ can look up the corresponding β_i value. Otherwise, $\mathcal{C}$ creates a new entry in H by uniformly sampling $\beta_i \in \{-1, 1\}$. $\mathcal{C}$ returns $\beta_1, \ldots, \beta_K$.

Aggregate Forgery: $\mathcal{A}$ forges an aggregate signature $(\boldsymbol{z}, \{\mu_i, \widehat{\boldsymbol{y}_i}|_\Omega\}_{i=1}^K)$, that verifies for $\{\widehat{\boldsymbol{f}_i}|_\Omega\}_{i=1}^K$ and messages $\{\mu_i\}_{i=1}^K$. If the forger fails so does $\mathcal{C}$ and returns `ABORT`.

Signature Extraction:

1. Challenger $\mathcal{C}$ retrieves $\boldsymbol{z}_i, \beta_i$ from $\mathsf{H}(\widehat{\boldsymbol{y}_i}|_\Omega, \mu_i, \widehat{\boldsymbol{f}_K}|_\Omega)$ for $i = 1, \ldots, N_S$.
2. Challenger $\mathcal{C}$ eliminates signatures obtained from the signing oracle for $\widehat{\boldsymbol{f}_K}|_\Omega$ by computing
$$\boldsymbol{z}' = \beta_K(\boldsymbol{z} - \sum_{i=1}^{N_S} \beta_i \boldsymbol{z}_i)$$
3. Recover signature by computing
$$\boldsymbol{z}_A = \boldsymbol{z}' - \boldsymbol{f}_{\mathcal{C}} \star \boldsymbol{c}_A$$
4. Return $(\boldsymbol{z}_A, \boldsymbol{c}_A, \mu_K)$ as the $\mathsf{PASS}_{\mathtt{RS}}$ signature.

Correctness. The challenger $\mathcal{C}$ returns $(\boldsymbol{z}_A, \boldsymbol{c}_A, \mu_K)$ as the $\mathsf{PASS}_{\mathtt{RS}}$ signature and succeeds only if $\widehat{\boldsymbol{f}_{N_H}}|_\Omega = \widehat{\boldsymbol{f}_K}|_\Omega$, i.e. the last hash query is performed with the challenge key. This happens with probability $1/N_H$. Otherwise the the simulation ends with an `ABORT`. For the remaining analysis we assume this to hold: $\widehat{\boldsymbol{f}_{N_H}}|_\Omega = \widehat{\boldsymbol{f}_K}|_\Omega$. To verify the signature we run the $\mathsf{PASS}_{\mathtt{RS}}$ verification process for instantiation with parameters (q, N, k', b'). There are two checks in the verification process: check of the norm of the signature $\boldsymbol{z}_A$ and the check on the recomputed challenge $\boldsymbol{c}$. Starting with the latter the challenge polynomial is recomputed from the signature and public key as follows. We first compute

$$\widehat{\boldsymbol{y}'_A}|_\Omega = \widehat{\boldsymbol{z}_A}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega \odot \widehat{\boldsymbol{c}_A}|_\Omega.$$

and then the challenge

$$\boldsymbol{c}' = \mathsf{FormatC}(\mathsf{Hash}(\widehat{\boldsymbol{y}'_A}|_\Omega, \mu_K, \widehat{\boldsymbol{f}_A}|_\Omega))$$

Then we check if the newly computed challenge matches the one received in the signature $\boldsymbol{c}' = \boldsymbol{c}_A$. Remember that

$$\boldsymbol{z}_A = \boldsymbol{z}' - \boldsymbol{f}_{\mathcal{C}} \star \boldsymbol{c}_A \quad \text{and} \quad \boldsymbol{z}' = \boldsymbol{z} - \sum_{i=1}^{N_S} \beta_i \boldsymbol{z}_i.$$

By construction the signatures returned during the signature queries satisfy

$$\boldsymbol{z}_i = \boldsymbol{f}_K \star \boldsymbol{c}_i + \boldsymbol{y}_i = (\boldsymbol{f}_\mathcal{C} + \boldsymbol{f}_A) \star \boldsymbol{c}_i + \boldsymbol{y}_i = \boldsymbol{f}_\mathcal{C} \star \boldsymbol{c}_i + \underbrace{\boldsymbol{f}_A \star \boldsymbol{c}_i + \boldsymbol{y}_i}_{\boldsymbol{y}_i^\mathcal{C}},$$

where $\boldsymbol{z}'$ denotes the remaining part of the aggregate signature with the signature queries removed,

$$\boldsymbol{z}' = \boldsymbol{z} - \sum_{i=1}^{N_S} \beta_i \boldsymbol{z}_i.$$

Since

$$\boldsymbol{z}' = \beta_K(\boldsymbol{f}_K \star \boldsymbol{c}_A + \boldsymbol{y}_K) + \sum_{i=1}^{N_H-1} \beta_i(\boldsymbol{f}_i \star \boldsymbol{c}_i + \boldsymbol{y}_i),$$

and since $\boldsymbol{y}_A$ was defined to satisfy

$$\beta_K \boldsymbol{y}_A = \beta_K \boldsymbol{y}_K + \sum_{i=1}^{N_H-1} \beta_i(\boldsymbol{f}_i \star \boldsymbol{c}_i + \boldsymbol{y}_i),$$

we can write

$$\beta_K \boldsymbol{z}' = (\boldsymbol{f}_K \star \boldsymbol{c}_A) + \boldsymbol{y}_A.$$

Let us recall that $\boldsymbol{f}_K = \boldsymbol{f}_\mathcal{C} + \boldsymbol{f}_A$. We expand the equation

$$\boldsymbol{z}' = \big((\boldsymbol{f}_\mathcal{C} + \boldsymbol{f}_A) \star \boldsymbol{c}_A\big) + \boldsymbol{y}_A = \boldsymbol{f}_\mathcal{C} \star \boldsymbol{c}_A + \boldsymbol{f}_A \star \boldsymbol{c}_A + \boldsymbol{y}_A.$$

Since we know $f_\mathcal{C} \star \boldsymbol{c}_A$, we can subtract it from the short version of the signature $\boldsymbol{z}'$:

$$\boldsymbol{z}_A = \boldsymbol{z}' - f_\mathcal{C} \star \boldsymbol{c}_A = \boldsymbol{f}_A \star \boldsymbol{c}_A + \boldsymbol{y}_A.$$

Now we have a valid signature $(\boldsymbol{z}_A, \boldsymbol{c}_A)$ for Alice for the message μ_K and public key $\widehat{\boldsymbol{f}_A}|_\Omega$. One can verify the signature by computing and comparing if the challenges $\boldsymbol{c}_A$ match:

$$\widehat{\boldsymbol{y}_A}|_\Omega = \widehat{\boldsymbol{z}_A}|_\Omega - \widehat{\boldsymbol{f}_A}|_\Omega \odot \widehat{\boldsymbol{c}_A}|_\Omega$$

This reveals

$$\boldsymbol{c}_A = \mathsf{Hash}(\widehat{\boldsymbol{y}_A}|_\Omega, \mu_K, \widehat{\boldsymbol{f}_A}|_\Omega)$$

In addition to the check on the challenge, the norm bound on the signature $\boldsymbol{z}_A$ also needs to be satisfied:

$$\|\boldsymbol{z}_A\| = \|\boldsymbol{z}' - \boldsymbol{f}_\mathcal{C}\boldsymbol{c}_A\| \leq \|\boldsymbol{z}'\| + \|\boldsymbol{f}_\mathcal{C}\boldsymbol{c}_A\| = \|\boldsymbol{z}'\| + b$$

and

$$\|\boldsymbol{z}'\| = \left\|\boldsymbol{z} - \sum_{i=1}^{N_S} \beta_i \boldsymbol{z}_i\right\| = \left\|\sum_{i=1}^{N_H} \boldsymbol{z}_i\right\| \leq N_H(k-b).$$

The Central Limit Theorem implies that $\|\boldsymbol{z}'\| \approx \sqrt{N_H}(k-b)$. To compensate for the infinity norm check, we add a constant $\mathcal{O}(1)$, and then $\boldsymbol{z}_A$ satisfies

$$\boldsymbol{z}_A \in \mathcal{B}^\infty\left(\mathcal{O}(1)\sqrt{N_H}(k-b)\right).$$

Efficiency. The computations during the simulation performed by the Challenger $\mathcal{C}$ are summarized as follows:

- **Challenge PK:** 1 $\mathcal{F}_\Omega$-transformation, 1 polynomial addition.
- **Signature Query:** 2 $\mathcal{F}_\Omega$-transformation, 2 addition/subtraction, 1 coefficient multiplication, 1 polynomial multiplication, 1 table insert for each signature query (N_S).
- **Message Hash Query:** 1 $\mathcal{F}_\Omega$-transformation, 1 coefficient multiplication and 1 coefficient addition for $N_H - 1$ queries. In addition 1 coefficient addition and 1 Hash calculation.
- **Single Extraction:** 1 subtraction for each N_S, 1 multiplication and 1 additional subtraction.

Hence in terms of primitive operations, the total computation time τ is given by

$$\tau = (2N_S + 1)\tau_{\mathcal{F}_\Omega} + (4N_S + 2N_H + 3)\tau_{\odot} + (N_S + 1)\tau_{\times} + (N_S + N_H)\tau_{\mathsf{H}} + \tau_{\mathsf{Hash}}.$$

B The Coordinate Moment of Points on a Sphere

In this section we compute the moments of the coordinates of points on a sphere, which is an elementary calculus exercise.

Lemma 2. *Let $S^{N-1} \subset \mathbb{R}^N$ denote the unit sphere taken with the uniform probability measure, and let X_i denote any one of the coordinate functions on $\mathbb{R}^N$. Then the odd moments of X_i are 0, and the moments of $|X_i|$ are given by*

$$E\big(|X_i|^\ell\big) = \frac{\Gamma\left(\frac{\ell+1}{2}\right)\Gamma\left(\frac{N}{2}\right)}{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{N+\ell}{2}\right)}$$

In particular, the variance is $E(X_i^2) = 1/N$.

Proof. The sphere is invariant under $X_i \to -X_i$, so it is clear that the odd moments of X_i vanish. For the moments of $|X_i|$ we compute

$$\begin{aligned}
&\int_{S^{N-1}} |X_1|^\ell \, dV_{S^{N-1}} \\
&= \int_0^\pi \cdots \int_0^\pi \int_0^{2\pi} |\cos\theta_1|^\ell \prod_{i=1}^{N-2} \sin^{N-1-i}\theta_i \, d\theta_1 \cdots d\theta_{N-1} \\
&= \left(\int_0^\pi |\cos\theta_1|^\ell \cdot \sin^{N-2}\theta_1 \, d\theta_1\right)\left(\prod_{i=2}^{N-2}\int_0^\pi \sin^{N-1-i}\theta_i \, d\theta_i\right)\left(\int_0^{2\pi} d\theta_{N-1}\right) \\
&= \left(2\int_0^{\pi/2} \cos^\ell\theta_1 \cdot \sin^{N-2}\theta_1 \, d\theta_1\right)\left(\prod_{i=2}^{N-2} 2\int_0^{\pi/2} \sin^{N-1-i}\theta_i \, d\theta_i\right)\left(4\int_0^{\pi/2} d\theta_{N-1}\right)
\end{aligned}$$

$$= B\left(\frac{\ell+1}{2}, \frac{N-1}{2}\right) \cdot \prod_{i=2}^{N-2} B\left(\frac{1}{2}, \frac{N-i}{2}\right) \cdot 2B\left(\frac{1}{2}, \frac{1}{2}\right)$$
$$= \frac{\Gamma\left(\frac{\ell+1}{2}\right)\Gamma\left(\frac{N-1}{2}\right)}{\Gamma\left(\frac{N+\ell}{2}\right)} \cdot \prod_{i=2}^{N-2} \frac{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{N-i}{2}\right)}{\Gamma\left(\frac{N-i+1}{2}\right)} \cdot 2\frac{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{1}{2}\right)}{\Gamma(1)}$$
$$= 2\Gamma\left(\frac{1}{2}\right)^{N-1} \Gamma\left(\frac{\ell+1}{2}\right) \Gamma\left(\frac{N+\ell}{2}\right)^{-1}.$$

In particular, taking $\ell = 0$ gives the well-known formula

$$\mathrm{Vol}(S^{N-1}) = 2\Gamma(1/2)^N/\Gamma(N/2) \tag{45}$$

for the volume of the sphere, and thus the probability measure on S^{N-1} is $dV_{S^{N-1}}$ divided by (45). Hence

$$E\big(|X_i|^\ell\big) = \mathrm{Vol}(S^{N-1})^{-1} \int_{S^{N-1}} |X_1|^\ell \, dV_{S^{N-1}} = \frac{\Gamma\left(\frac{\ell+1}{2}\right)\Gamma\left(\frac{N}{2}\right)}{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{N+\ell}{2}\right)},$$

which is the desired formula.

To compute the variance, we use $\Gamma(z+1) = z\Gamma(z)$ to compute

$$E(X_i^2) = \frac{\Gamma(3/2)\Gamma(N/2)}{\Gamma(1/2)\Gamma(N/2+1)} = \frac{(1/2)\Gamma(1/2)\Gamma(N/2)}{\Gamma(1/2)(N/2)\Gamma(N/2)} = \frac{1}{N}.$$

More generally if ℓ is even, one can expand the gamma functions to get

$$E(X_i^\ell) = \prod_{j=0}^{\ell/2-1} \frac{2j+1}{N+2j},$$

so for example, $E(X_1^4) = 3/(N^2+2N)$ and $E(X_1^6) = 15/(N^3+6N^2+8N)$.

C Chen and Nguyen's BKZ-Simulator

In this section we describe that BKZ-simulator of Chen and Nguyen [13]. It predicts the quality of the output from running BKZ β for a given number of rounds using a given, reasonably large, blocksize β. We sumamrize their results, using slightly different notation.

Theorem 5 (BKZ-Simulator [13]).

Input:

$\ell_1, \ldots, \ell_N$	*the logarithms of the lengths of the vectors in the Gram-Schmidt orthogonalization of the initial basis for the lattice* Λ.
BlockSize	*the blocksize used by BKZ, must be* ≥ 45.
Rounds	*the number of rounds (iterations) of BKZ.*

Output:

$\ell'_1, \ldots, \ell'_N$	*the logarithms of the lengths of the vectors in the Gram-Schmidt orthogonalization of the basis determined by BKZ.*

Run Time:

The expected RunTime *of BKZ is given by the formula*

$$\log_2(\mathsf{RunTime}) = \log_2(\mathsf{Rounds} \cdot N) + 0.64 \cdot \mathsf{BlockSize} - 20.356.$$

N.B. This run-time estimate from [13] *and* [26] *is superceded by the far more conservative estimate given by* (14) *in Proposition* 4.

Proof. See Algorithm 10 in Table 17, as well as [13, Algorithm 2] and the subsequent analysis. Following [13], we estimate the run time of BKZ as follows: Each round consists of roughly $N = \dim \Lambda$ enumerations. The cost of each enumeration is equal to the number of enumeration tree nodes visited multiplied by the cost of visiting one node. Hence

$$\mathsf{RunTime} \approx (\text{number of rounds}) \cdot N \cdot (\text{nodes per enumeration}) \cdot (\text{cost per node}).$$

In [13] the cost per node is given as roughly 200 cycles. We estimate the number of nodes per enumeration using the equation

$$\log_2(\text{nodes per enumeration}) = 0.64 \cdot \mathsf{BlockSize} - 28,$$

which is derived in Lepoint et al. [26]. Hence

$$\log_2(\mathsf{RunTime}) = \log_2(\mathsf{Rounds} \cdot N) + \underbrace{(0.64 \cdot \mathsf{BlockSize} - 28)}_{\text{enumeration cost}} + \underbrace{\log_2(200)}_{\text{cost per node}}.$$

Then $-28 + \log_2(200) = -20.356$ gives the stated formula

Remark 15. Algorithm 10 in Table 17 describes our implementation of [13, Algorithm 2], the Chen–Nguyen BKZ Simulator. We briefly indicate the differences between our implementation and [13, Algorithm 2].

- We have renamed some of the variables. In particular, the dimension of the lattice is N, and the blocksize and number of rounds are denoted by BlockSize and Rounds.
- We explicitly include the initialization $\ell'_i = \ell_i$ for $1 \leq i \leq N$. We note that some initial assignment is required, since otherwise an unassigned ℓ'_i value might be needed at Step 17.

- We initialize the quantity logV in Step 9 and then update the value of logV in Steps 14 and 17. This updating procedure replaces the following assignment statement from [13, Algorithm 2]:

$$\log \mathrm{V} \leftarrow \sum_{i=1}^{\min\{f,N\}} \ell_i - \sum_{i=1}^{k-1} \ell'_i.$$

  We have thus replaced $2k$ additions with 2 additions, and since the k loop runs from 1 to roughly N, the resulting run-time of the simulator is greatly reduced.

D Average L_2-Norm of $F \star c$

The following calculation may be useful for future work, but is not used in this paper.

Lemma 3. *Let $\boldsymbol{F} \in \mathbb{Z}^N$. Then for all $N \geq 1$ and $N/2 \geq d \geq 1$,*

$$\operatorname*{Mean}_{\boldsymbol{c} \leftarrow \mathcal{T}_N(d)} \left(\|\boldsymbol{F} \star \boldsymbol{c}\|_2^2 \right) = 2d \, \|\boldsymbol{F}\|_2^2 .$$

Hence for a randomly chosen $\boldsymbol{c} \in \mathcal{T}_N(d)$, one expects

$$\|\boldsymbol{F} \star \boldsymbol{c}\|_2 \approx \sqrt{2d} \cdot \|\boldsymbol{F}\|_2 , \tag{46}$$

which accords with viewing the coefficients of $\boldsymbol{F} \star \boldsymbol{c}$ as summing a $2d_c$-step random walk through the coefficients of $\boldsymbol{F}$.

Proof. To ease notation, we write $\mathcal{T}$ for $\mathcal{T}_N(d)$. Then

$$\begin{aligned}
\operatorname*{Mean}_{\boldsymbol{c} \leftarrow \mathcal{T}_N(d)} \left(\|\boldsymbol{F} \star \boldsymbol{c}\|_2^2 \right) &= (\#\mathcal{T})^{-1} \sum_{\boldsymbol{c} \in \mathcal{T}} \|\boldsymbol{F} \star \boldsymbol{c}\|_2^2 \\
&= (\#\mathcal{T})^{-1} \sum_{\boldsymbol{c} \in \mathcal{T}} \sum_{k=0}^{N-1} (\boldsymbol{F} \star \boldsymbol{c})_k^2 \\
&= (\#\mathcal{T})^{-1} \sum_{\boldsymbol{c} \in \mathcal{T}} \sum_{k=0}^{N-1} \left(\sum_{i=0}^{N-1} F_{k-i} c_i \right)^2 \\
&= (\#\mathcal{T})^{-1} \sum_{\boldsymbol{c} \in \mathcal{T}} \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} F_{k-i} c_i F_{k-j} c_j \\
&= (\#\mathcal{T})^{-1} \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} F_{k-i} F_{k-j} \sum_{\boldsymbol{c} \in \mathcal{T}} c_i c_j .
\end{aligned}$$

Algorithm 10 BKZ Simulator

Input: $N =$ lattice dimension

$\ell_1, \ldots, \ell_N$, the logarithmic Gram–Schmidt norms of the initial lattice basis, i.e., if $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ is the initial basis, then $\ell_i = \log \|\boldsymbol{v}_i^*\|_2$, where $\boldsymbol{v}_i^*$ is given by (12)

$\mathsf{BlockSize} =$ BKZ-blocksize, must satisfy $\mathsf{BlockSize} \geq 45$

$\mathsf{Rounds} =$ number of BKZ-rounds

$r_1, \ldots, r_{45}$, average logarithmic Gram–Schmidt norms of HKZ-reduced random unit-volume 45-dimensional lattices (determined experimentally, see Table 18)

Output: $\ell'_1, \ldots, \ell'_N$, the logarithmic Gram–Schmidt norms of the expected output lattice basis after the specified number of Rounds of BKZ-reduction at the specified $\mathsf{BlockSize}$. The associated Hermite Root Factor (HRF) δ is given by

$$\log \delta = \frac{1}{N-1}\left(\ell'_1 - \frac{1}{N}\sum_{i=1}^{N} \ell'_i\right).$$

for $i := 1$ **to** N **step** 1 **do**
  $\ell'_i = \ell_i$
end for
for $d := 46$ **to** $\mathsf{BlockSize}$ **step** 1 **do**
  $c_d \leftarrow \log\left(\Gamma(d/2+1)^{1/d}/\pi^{1/2}\right) = \log\big(\mathrm{GH}(\mathbb{Z}^d)\big)$
end for
for $j := 1$ **to** Rounds **step** 1 **do**
  $\phi \leftarrow$ true // flag to store whether $L_{[k,N]}$ has changed
  $\mathrm{logV} \leftarrow \sum_{i=1}^{\beta} \ell_i$
  for $k := 1$ **to** $N - 45$ **step** 1 **do**
    $d \leftarrow \min(\mathsf{BlockSize}, N-k+1)$ // dimension of local block
    $f \leftarrow k + \mathsf{BlockSize}$ // end index of local block (if $\leq N$)
    if $f \leq N$ **then**
      $\mathrm{logV} \leftarrow \mathrm{logV} + \ell_f$
    end if
    if $k \geq 2$ **then**
      $\mathrm{logV} \leftarrow \mathrm{logV} - \ell'_{k-1}$
    end if // at this step, logV equals $\sum_{i=1}^{\min\{f,N\}} \ell_i - \sum_{i=1}^{k-1} \ell'_i$
    if $\phi =$ true **then**
      if $d^{-1}\,\mathrm{logV} + c_d < \ell_k$ **then**
        $\ell'_k \leftarrow d^{-1}\,\mathrm{logV} + c_d$
        $\phi \leftarrow$ false
      end if
    else
      $\ell'_k \leftarrow d^{-1}\,\mathrm{logV} + c_d$
    end if
  end for
  $\mathrm{logV} \leftarrow \sum_{i=1}^{N} \ell_i - \sum_{i=1}^{N-45} \ell'_i$
  for $k := N - 44$ **to** N **step** 1 **do**
    $\ell'_k \leftarrow 45^{-1}\,\mathrm{logV} + r_{k+45-N}$
  end for
  for $k := 1$ **to** N **step** 1 **do**
    $\ell_k \leftarrow \ell'_k$
  end for
end for

Table 17: BKZ Simulator - Optimized Version of [13, Algorithm 2]

0.789528,	0.780003,	0.750872,	0.706520,	0.696345,	0.660534,
0.626275,	0.581481,	0.553171,	0.520811,	0.487994,	0.459541,
0.414638,	0.392812,	0.339090,	0.306561,	0.276041,	0.236699,
0.196186,	0.161214,	0.110895,	0.067826,	0.027281,	−0.023461,
−0.032053,	−0.094033,	−0.129109,	−0.176965,	−0.209406,	−0.265868,
−0.299031,	−0.349339,	−0.380428,	−0.427399,	−0.474945,	−0.530141,
−0.561625,	−0.612009,	−0.669011,	−0.713767,	−0.754042,	−0.808610,
−0.859933,	−0.884480,	−0.886667			

Table 18: Average logarithmic Gram–Schmidt norms of HKZ-reduced random unit-volume 45-dimensional lattices, used by the BKZ Simulator (Algorithm 10)

In order to compute the inner sum, we consider two cases. First, if $i \neq j$, we momentarily write $\boldsymbol{c} \to \boldsymbol{c}'$ for the permutation of $\mathcal{T}$ that flips the sign of the ith coordinate of $\boldsymbol{c}$. Then

$$\sum_{\boldsymbol{c}\in\mathcal{T}} c_i c_j = \sum_{\substack{\boldsymbol{c}\in\mathcal{T}\\ c_i=1}} c_j - \sum_{\substack{\boldsymbol{c}\in\mathcal{T}\\ c_i=-1}} c_j = \sum_{\substack{\boldsymbol{c}\in\mathcal{T}\\ c_i=1}} c_j - \sum_{\substack{\boldsymbol{c}'\in\mathcal{T}\\ c'_i=1}} c'_j = \sum_{\substack{\boldsymbol{c}\in\mathcal{T}\\ c_i=1}} c_j - \sum_{\substack{\boldsymbol{c}'\in\mathcal{T}\\ c'_i=1}} c_j = 0,$$

where in the penultimate equality we have used the fact that $i \neq j$ to conclude that $c'_j = c_j$. Second, if $i = j$, then

$$\sum_{\boldsymbol{c}\in\mathcal{T}} c_i^2 = \#\{\boldsymbol{c}\in\mathcal{T} : c_i = \pm 1\} = 2\#\{\boldsymbol{c}\in\mathcal{T} : c_i = 1\} = 2\binom{N-1}{d-1,d} = \frac{2d}{N}\binom{N}{d,d}.$$

Subtituting and using $\#\mathcal{T} = \binom{N}{d,d}$ yields

$$\operatorname*{Mean}_{\boldsymbol{c}\leftarrow\mathcal{T}_N(d)}\left(\|\boldsymbol{F}\star\boldsymbol{c}\|_2^2\right) = \#\mathcal{T}^{-1}\sum_{k=0}^{N-1}\sum_{i=0}^{N-1} F_{k-i}^2\,\frac{2d}{N}\binom{N}{d,d} = N\cdot\|\boldsymbol{F}\|_2^2\cdot\frac{2d}{N}.$$

which completes the proof of the lemma.

References

1. M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *In Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 99–108. ACM, 1996.
2. M. R. Albrecht, L. Ducas, G. Herold, E. Kirshanova, E. W. Postlethwaite, and M. Stevens. The general sieve kernel and new records in lattice reduction. Cryptology ePrint Archive, Report 2019/089, 2019. https://eprint.iacr.org/2019/089.
3. M. R. Albrecht, R. Fitzpatrick, and F. Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In *Information security and cryptology—ICISC 2013*, volume 8565 of *Lecture Notes in Comput. Sci.*, pages 293–310. Springer, Cham, 2014.

4. M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer. Revisiting the expected cost of solving uSVP and applications to LWE. In *Advances in cryptology—ASIACRYPT 2017. Part I*, volume 10624 of *Lecture Notes in Comput. Sci.*, pages 297–322. Springer, Cham, 2017.
5. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343, 2016.
6. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange: A new hope. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 327–343, USA, 2016. USENIX Association.
7. R. E. Bansarkhani and J. Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In *15th International Conference on Cryptology and Network Security - CANS 2016*, November 2016.
8. D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *ASIACRYPT*, 2018.
9. D. Boneh and D. Freeman. Homomorphic signatures for polynomial functions, 2011.
10. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
11. M. Braithwaite. Experimenting with post-quantum cryptography, 2016. Google Security Post.
12. Y. Chen. *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*. PhD thesis, Paris 7, 2013. 133 p.
13. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT 2011*, pages 1–20. Springer, 2011.
14. Y. Doroz, J. Hoffstein, J. H. Silverman, and B. Sunar. Rejction sampling and quantifiable transcript security for identification-based signature schemes, 2020. preprint.
15. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013.
16. W. Feller. *An introduction to probability theory and its applications. Vol. II*. Second edition. John Wiley & Sons, Inc., New York-London-Sydney, 1971.
17. N. Gama and P. Q. Nguyen. Predicting lattice reduction. In N. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, 2008.
18. F. Göpfert. *Securely Instantiating Cryptographic Schemes Based on the Learning with Errors Assumption*. PhD thesis, Technische Universität Darmstadt, 2016. http://tuprints.ulb.tu-darmstadt.de/5850/.
19. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 469–477, New York, NY, USA, 2015. ACM.
20. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In E. Prouff and P. Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, 2012.
21. J. Hoffstein, J. Pipher, J. Schanck, J. Silverman, and W. Whyte. Pass-rs: Practical signatures from the partial fourier recovery problem. In I. Boureanu, P. Owesarski, and S. Vaudenay, editors, *ACNS*, volume 8479 of *LNCS*, page 476–493. Springer, 2014.

22. J. Hoffstein and J. H. Silverman. Polynomial rings and efficient public key authentication II. In K.-Y. Lam, I. Shparlinski, H. Wang, and C. Xing, editors, *Cryptography and Computational Number Theory*, volume 20 of *Progress in Computer Science and Applied Logic*, pages 269–286. Birkhäuser, 2001.
23. A. Hülsing, J. Rijneveld, J. M. Schanck, and P. Schwabe. High-speed key encapsulation from ntru. Cryptology ePrint Archive, Report 2017/667, 2017. https://eprint.iacr.org/2017/667.
24. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. In *NEC Research and Development*, page 1–8. 1983.
25. A. Langley. CECPQ2, 2018.
26. T. Lepoint and M. Naehrig. A comparison of the homomorphic encryption schemes fv and yashe. In D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, pages 318–335, Cham, 2014. Springer International Publishing.
27. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, pages 598–616. Springer, 2009.
28. V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.
29. G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018. Accessed:2018-01-22.
30. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, pages 245–254, New York, NY, USA, 2001. ACM.
31. D. Micciancio. Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169, 2004. Preliminary version in STOC 2002.
32. D. Micciancio and C. Peikert. Hardness of sis and lwe with small parameters. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 21–39, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
33. D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, Apr. 2007.
34. M. E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, 2(4):19–20, 1959.
35. P. Q. Nguyen. Hermite's constant and lattice algorithms. In *The LLL algorithm*, pages 19–69. Springer, 2009.
36. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.
37. C. P. Schnorr. Lattice reduction by random sampling and birthday methods. In *STACS 2003*, volume 2607 of *Lecture Notes in Comput. Sci.*, pages 145–156. Springer, Berlin, 2003.
38. J. Shao. *Mathematical statistics*. Springer Texts in Statistics. Springer-Verlag, New York, second edition, 2003.
39. M. Sibuya. A method for generating uniformly distributed points on N-dimensional spheres. *Ann. Inst. Statist. Math.*, 14:81–85, 1962.
40. Wikipedia contributors. Volume of an n-ball — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Volume_of_an_n-ball&oldid=950347574, 2020. [Online; accessed 3-May-2020].

41. Y. Yu and L. Ducas. Second order statistical behavior of LLL and BKZ. In *Selected areas in cryptography—SAC 2017*, volume 10719 of *Lecture Notes in Comput. Sci.*, pages 3–22. Springer, Cham, 2018.

E Tables of MMSAT and $\mathsf{PASS}_{\mathtt{RS}}$ Parameters Secure Against Known Attacks

In this section we give sample parameters for MMSAT and $\mathsf{PASS}_{\mathtt{RS}}$ that are secure against the known attacks described in this paper. We recall that the Hermite root factors listed in the tables have the following meanings, and we refer to Proposition (10) for the explicit formula used to compute each HRF.

$$\begin{aligned}
\delta_\beta &= \text{HRF achieved by BKZ-}\beta\text{ for blocksize }\beta = \lambda/0.265.\\
\delta_{\text{forge-}\boldsymbol{z}} &= \text{HRF required to forge an amalgamate signature } \boldsymbol{z}.\\
\delta_{\text{fake-}\boldsymbol{F}} &= \text{HRF required to find an } \boldsymbol{F} \text{ that serves a a fake signing key.}\\
\delta_{\text{USVP-08}} &= \text{2008-estimate of HRF required to solve the UniqueSVP and find } \boldsymbol{f}.\\
\delta_{\text{USVP-16}} &= \text{2016-esimtate of HRF required to solve the UniqueSVP and find } \boldsymbol{f}.
\end{aligned}$$

K	N	q	t	d_f	d_c	k	b	β	δ_{BKZ}	δ_{forge}	$\delta_{\text{fake-}\boldsymbol{F}}$	$\delta_{\text{USVP-08}}$	$\delta_{\text{USVP-16}}$	1 Key (Bytes)	1 Sig (Bytes)	Agg-Sig (KByte)
$\lambda = 128, \quad B_k = 3, \quad B_q = 1/4, \quad t \approx 1/3 \cdot N, \quad \text{Prob(Accept)} = 25.00\%$																
1000	4111	$2^{24.909}$	1370	1370	14	83047	28	484	1.0034887	1.0033051	1.0028452	1.0013979	1.0009122	4266	8398	49
5000	4271	$2^{26.125}$	1423	1370	14	86279	28	484	1.0034887	1.0033179	1.0028734	1.0014155	1.0009585	4647	8754	178
10000	4349	$2^{26.651}$	1449	1370	14	87854	28	484	1.0034887	1.0033165	1.0028792	1.0014202	1.0009750	4828	8928	339
$\lambda = 128, \quad B_k = 3, \quad B_q = 1/4, \quad t \approx 1/3 \cdot N, \quad \text{Prob(Accept)} = 99.00\%$																
1000	5119	$2^{32.333}$	1706	1706	14	14261428	28	484	1.0034887	1.0033471	1.0029693	1.0014580	1.0011042	6896	15208	59
5000	5297	$2^{33.544}$	1765	1706	14	14757332	28	484	1.0034887	1.0033439	1.0029775	1.0014648	1.0011270	7401	15769	189
10000	5381	$2^{34.066}$	1793	1706	14	14991354	28	484	1.0034887	1.0033381	1.0029769	1.0014658	1.0011348	7636	16034	350
$\lambda = 128, \quad B_k = 3, \quad B_q = 1/4, \quad t \approx 1/2 \cdot N, \quad \text{Prob(Accept)} = 25.00\%$																
1000	3217	$2^{24.655}$	1608	1072	15	69632	30	484	1.0034887	1.0032624	1.0026798	1.0026555	1.0025171	4956	6469	47
5000	3343	$2^{25.871}$	1671	1072	15	72359	30	484	1.0034887	1.0032715	1.0027084	1.0026874	1.0025694	5404	6746	176
10000	3407	$2^{26.398}$	1703	1072	15	73744	30	484	1.0034887	1.0032666	1.0027129	1.0026935	1.0025810	5620	6887	337
$\lambda = 256, \quad B_k = 3, \quad B_q = 1/4, \quad t \approx 1/3 \cdot N, \quad \text{Prob(Accept)} = 25.00\%$																
1000	7393	$2^{26.806}$	2464	2464	29	309338	58	967	1.0020973	1.0019948	1.0016931	1.0008364	1.0005421	8257	16855	96
5000	7643	$2^{27.965}$	2547	2464	28	308771	56	967	1.0020973	1.0020019	1.0017115	1.0008462	1.0005680	8904	17423	354
10000	7759	$2^{28.486}$	2586	2464	28	313456	56	967	1.0020973	1.0020040	1.0017175	1.0008502	1.0005788	9209	17708	675
$\lambda = 256, \quad B_k = 3, \quad B_q = 1/4, \quad t \approx 1/3 \cdot N, \quad \text{Prob(Accept)} = 99.00\%$																
1000	9041	$2^{34.102}$	3013	3013	27	48576911	54	967	1.0020973	1.0020157	1.0017684	1.0008704	1.0006535	12844	28857	114
5000	9311	$2^{35.305}$	3103	3013	27	50027609	54	967	1.0020973	1.0020186	1.0017779	1.0008766	1.0006693	13694	29768	373
10000	9431	$2^{35.824}$	3143	3013	27	50672363	54	967	1.0020973	1.0020190	1.0017811	1.0008788	1.0006754	14075	30173	694
$\lambda = 256, \quad B_k = 3, \quad B_q = 1/4, \quad t \approx 1/2 \cdot N, \quad \text{Prob(Accept)} = 25.00\%$																
1000	5791	$2^{26.503}$	2895	1930	30	250669	60	967	1.0020973	1.0019722	1.0015930	1.0015851	1.0015437	9591	12983	92
5000	6007	$2^{27.717}$	3003	1930	30	260018	60	967	1.0020973	1.0019744	1.0016075	1.0016013	1.0015697	10405	13507	350
10000	6089	$2^{28.236}$	3044	1930	30	263567	60	967	1.0020973	1.0019785	1.0016161	1.0016104	1.0015837	10744	13706	671

Table 19: Sample Parameters for MMSAT

N	q	t	d_f	d_c	k	b	β	δ_{BKZ}	δ_{forge}	$\delta_{\text{fake-}\boldsymbol{F}}$	$\delta_{\text{USVP-08}}$	$\delta_{\text{USVP-16}}$	Key (Bytes)	Sig (Bytes)
$\lambda = 128$ $B_q = 1/4$, $t \approx 1/3 \cdot N$, Prob(Accept) = 25.00%														
3163	$2^{18.062}$	1054	1054	15	68464	30	484	1.0034887	1.0032523	1.0026922	1.0013165	1.0005851	2380	6351
$\lambda = 128$ $B_q = 1/4$, $t \approx 1/3 \cdot N$, Prob(Accept) = 99.00%														
4201	$2^{25.480}$	1400	1400	14	11703902	28	484	1.0034887	1.0032998	1.0028750	1.0013994	1.0009276	4460	12331
$\lambda = 128$ $B_q = 1/4$, $t \approx 1/2 \cdot N$, Prob(Accept) = 25.00%														
2503	$2^{17.818}$	1251	834	16	57793	32	484	1.0034887	1.0031945	1.0024908	1.0024651	1.0021421	2787	4949
$\lambda = 256$ $B_q = 1/4$, $t \approx 1/3 \cdot N$, Prob(Accept) = 25.00%														
5807	$2^{19.939}$	1935	1935	30	251362	60	967	1.0020973	1.0019719	1.0016102	1.0007914	1.0003541	4823	13022
$\lambda = 256$ $B_q = 1/4$, $t \approx 1/3 \cdot N$, Prob(Accept) = 99.00%														
7507	$2^{27.369}$	2502	2502	29	43322562	58	967	1.0020973	1.0020002	1.0017167	1.0008411	1.0005543	8560	23806
$\lambda = 256$ $B_q = 1/4$, $t \approx 1/2 \cdot N$, Prob(Accept) = 25.00%														
4621	$2^{19.703}$	2310	1540	32	213366	64	967	1.0020973	1.0019365	1.0014842	1.0014761	1.0013379	5690	10226

Table 20: Sample Parameters for $\mathsf{PASS}_{\mathtt{RS}}$