DIGITAL SIGNATURE ALGORITHMS EHTV3 AND EHTV4 SUBMISSION TO NIST PQC

Principal Submitter: Igor Semaev.

Email Address: igor@ii.uib.no

Phone number: Department of Informatics reception (+47) 55584200.

Postal Address: Igor Semaev, Department of Informatics, University of Bergen, Postbox 7803, 5020 Bergen, Norway.

Auxiliary Submitter: Martin Feussner.

Inventor: Igor Semaev.

Developer: Martin Feussner.

Signature: See the printed version of the cover sheet.

Contents

Table 1. EHTv3 and EHTv4 Performance

EHT version -NIST category	v3-1	v3-3	v3-5	v4-1	v4-5
Signature (bytes)	169	255	344	369	857
Private Key (bytes)	368	532	701	419	925
Public Key (Kbytes)	83.5	191.6	349.0	1.11	2.63
Key Generation (msec)	194	597	1530	12.1	115
Signature Generation (msec)	75.8	206	305	9.0	59.3
Signature Verification (msec)	0.82	1.78	3.16	3.85	26.2
# trials for a signature	2.6	3.22	2.01	4.97	3.46

1. **Introduction**

Digital signature algorithm EHTv3 has some similarity with the public key crypto-system EHT in [5]. However, it is impossible to use this crypto-system directly to generate signatures as the length of the cipher-text is larger than the length of the plain-text. Nevertheless, we call the new digital signature algorithm EHT too. The first version of the digital signature algorithm EHT, let's call that EHTv1, is in [18]. The second version EHTv2 appeared in the proceedings of NISK 2022, [19]. The current version EHTv3 mostly differs in the choice of the matrix C. Also, we submit EHTv4 which is very similar to EHTv3 but the arithmetic is in a finite group ring over $\mathbb{Z}_q$ instead of $\mathbb{Z}_q$ itself. That provides on the whole with a more efficient algorithm for comparable security levels at the expense of a larger signature size.

In Table 1 we summarised the performance of EHTv3 and EHTv4 signature algorithms for parameters in Sections 6 and 10 which fit different security levels according to the NIST call [14]. The average time estimates were calculated for 10^3 EHTv3 iterations and 10^4 EHTv4 iterations on a common computer with Windows 10 64-bit operating system and x64-based processor: 12thGen Intel(R) Core(TM) i7-12800H@2.40 GHz with 16.0 GB Ram.

The submitters acknowledge a number of useful discussions with Erik Mårtensson regarding the algorithm implementations.

1.1. q-**ary Lattices.**

Let q, n, m, s, such that $s < (q-1)/2$ and $n < m$, be positive integers. Suppose A is an $(m \times n)$-matrix over integers $\mathbb{Z}$ and L is a lattice generated by the columns of A modulo q. So, any $y \in L$ satisfies $y \equiv Ax \mod q$ for some $x \in \mathbb{Z}^n$. One says that L is a q-ary lattice of rank m. Let $e = (e_1, \ldots, e_m) \in \mathbb{Z}^m$, then $\max(e) = \max_{1 \le i \le m} |e_i|$ is the max norm of e. Also, we say $\max_l(e) \le s$ if at least l entries of e are at most s in absolute value.

Let's consider the following problem. Given $h \in \mathbb{Z}^m$, find $y \in L$ where $\max_l(h - y) \le s$ if such y does exist. That generally seems a hard problem. For instance, if $l = m$, then the solution to the problem implies a solution to the standard approximate Closest Vector Problem (in the Euclidean norm) for L with an approximation factor $O(\frac{s}{q^{1-n/m}})$. If the approximation factor is small, then the problem is considered hard in general lattices according to [13].

1.2. **Design Rationale.**

Let $A \in \mathbb{Z}_q^{m \times n}$ be a public matrix and let $h = \mathrm{HASH}(M) \in \mathbb{Z}_q^m$ be a hash value for a message M. The signature for M is $x \in \mathbb{Z}_q^n$ such that $h \equiv Ax + e \mod q$ and $e \in \mathbb{Z}^m$ satisfies $\max_l(e) \le s$. To forge a signature for M

one must compute $x \in \mathbb{Z}_q^n$ such that $\max_l(h - Ax \mod q) \le s$. That is an instance of the problem above.

We construct A with a trapdoor in order to implement the idea. Let k be a positive integer such that $n < m < kn$. There exists a positive integer $c < q$ and a matrix $T \in \mathbb{Z}_q^{kn\times n}$ with the following property. For every $a \in \mathbb{Z}_q^{kn}$, it is easy to compute $y \in \mathbb{Z}_q^n$ and $z \in \mathbb{Z}^{kn}$, and such that $a \equiv Ty + z \mod q$ and $\max(z) \le c$, that is every entry of z is bounded by c in absolute value. The definition of the matrix T is below, see (2) and Theorem 1.

We hide T by multiplying on the left by a matrix $C \in \mathbb{Z}^{m\times kn}$, every row of which is of 1-norm λ for a relatively small parameter λ. That means that the sum of absolute values of the entries in each row of C is λ. The choice of λ defines l and s. Also, one multiplies on the right by $B^{-1} \mod q$ for an arbitrary matrix $B \in \mathbb{Z}_q^{n\times n}$, invertible modulo q. The matrix $A \equiv CTB^{-1} \mod q$ is public and the matrices C, T, B are secret.

1.3. **Inequalities.** Let q, k, c be positive integers.

1.3.1. *Diagonal Tuples.* We call the tuple of integers $[t_1, t_2, \dots, t_k]$ diagonal if its entries are non-zero residues modulo q and at least one is coprime to q. Besides, for any integers $b_1, b_2, \dots, b_k$ there is an integer u such that

$$\begin{aligned} |(b_1 - t_1\, u) \bmod q| &\le c,\\ |(b_2 - t_2\, u) \bmod q| &\le c,\\ &\dots,\\ |(b_k - t_k\, u) \bmod q| &\le c, \end{aligned} \tag{1}$$

where the residues modulo q are taken smallest in absolute values. For relatively small q and k used in this work all such tuples may be found by brute force. Let, for instance, $q = 61, k = 3, c = 8$. There is only one tuple $[t_1, t_2, t_3] = [1, 4, 15]$ modulo q up to a permutation of entries, multiplication the tuple by a residue coprime to q and changing the sign of the entries such that for any integers b_1, b_2, b_3 the system of inequalities $|(b_1 - t_1u) \bmod 61| \le 8, |(b_2 - t_2u) \bmod 61| \le 8, |(b_k - t_3u) \bmod 61| \le 8$ has a solution u. For $q = 47, k = 2, c = 3$ there is only one such tuple $[t_1, t_2] = [1, 7]$. The solution of (1) is easy in this case, see Section 1.3.2 below.

1.3.2. *Solving the Inequalities.* Let $k = 2$, $q \le (2c+1)^2$ and $[t_1, t_2] = [1, t]$, where $t = 2c + 1$. In that case we show how to solve (1) efficiently. The system (1) is equivalent to $b_1 \equiv u + z_1, b_2 \equiv tu + z_2$, where $|z_1|, |z_2| \le c$ and therefore to $u \equiv b_1 - z_1, b_2 - tb_1 \equiv z_2 - tz_1$. For any b_1, b_2 that has a solution u such that $|z_1|, |z_2| \le c$ with the following algorithm.

(1) Set $a \equiv b_2 - tb_1 \mod q$, where $|a| \le q/2$.
(2) Expand a to the base t and get $a = y_2 + ty_1$, where $|y_2|, |y_1| < t$ and $\mathrm{sign}(a) = \mathrm{sign}(y_2) = \mathrm{sign}(y_1)$.
(3) If $|y_2| \le c$, then $z_2 = y_2$ and $z_1 = -y_1$. If $|y_2| > c$, then $z_2 = y_2 - \mathrm{sign}(a)t$ and $z_1 = -(y_1 + \mathrm{sign}(a))$.
(4) Set $u \equiv b_1 - z_1 \mod q$.

In order to prove the algorithm, first suppose $|y_2| \le c$. Then $z_1 = -y_1 = -(a - y_2)/t$. So, $|z_1| \le |a|/t \le c + 1/2$ as $|a| \le (2c+1)^2/2$. Since z_1 is an integer, we have $|z_1| \le c$.

Suppose $|y_2| > c$. As $y_1 = (a - y_2)/t$, we have $|y_1| \le (|a| - c - 1)/t < c$. One may represent $a = y_2 - \mathrm{sign}(a)t + t(y_1 + \mathrm{sign}(a))$ and set $z_2 = y_2 - \mathrm{sign}(a)t$

and $z_1 = -(y_1 + \mathrm{sign}(a))$. Obviously, $|z_2|, |z_1| \leq c$. Anyway, $a = z_2 - z_1 t$, where $|z_2|, |z_1| \leq c$.

2. EHTv3 Signature Algorithm

2.1. **Parameters.** Let $q, k, c, \lambda, \tau, s$ and n, m, l be positive integers, where the integers $q, k, c, \lambda, \tau, s$ are relatively small and $n < m < kn$, $s < (q-1)/2$. The integer q defines the arithmetic of the scheme. Also, let $h = \mathrm{HASH}(M)$ denote a hash value of the message M, encoded by a vector in $\mathbb{Z}_q^m$.

FIPS202_SHAKE256 was used to hash the message M and to obtain an output of byte length so that we have sufficient bytes for $h \in \mathbb{Z}_q^m$. For our proposed parameters, it is always greater than 64 bytes and ensures 256-bit security for h.

2.2. **Private Key.** The private key consists of three matrices C, T, B with entries in $\mathbb{Z}$ and $\mathbb{Z}_q$.

2.2.1. *Matrix T.* The matrix T is a $(kn \times n)$ -matrix as

$$T = \begin{pmatrix} t_{11} & 0 & \dots & 0 \\ t_{21} & 0 & \dots & 0 \\ t_{k1} & 0 & \dots & 0 \\ * & t_{12} & \dots & 0 \\ * & t_{22} & \dots & 0 \\ * & t_{k2} & \dots & 0 \\ * & * & \dots & t_{1n} \\ * & * & \dots & t_{2n} \\ * & * & \dots & t_{kn} \end{pmatrix}, \tag{2}$$

where the tuples $[t_{1j}, t_{2j}, \dots, t_{kj}]$ are diagonal, see Section 1.3.1. The entries of T below and to the left of the diagonal are denoted by $*$, they are secret and may be chosen randomly.

2.2.2. *Matrix C.* The matrix C is an $(m \times kn)$ -matrix over integers. The 1-norm of each row of C is λ. The matrix is constructed as a concatenation $C = (C_1|C_2)$. Let $P = P_1 + \ldots + P_\tau$ be a square $m \times m$ matrix, the sum of τ randomly chosen orthogonal permutation matrices $P_1, \dots, P_\tau$ (the permutations form a partial latin square). One randomly changes the signs of the non-zero entries of P and gets C_1. Experimentally, with a high probability C_1 is invertible over rationals. We assume that it is invertible modulo q as well. The matrix C_2 is of size $m \times (kn - m)$. The entries of C_2 are chosen randomly to provide that every row of C_2 is of 1-norm $\lambda - \tau$.

2.2.3. *Matrix B.* The matrix B is an arbitrary integer $(n \times n)$ -matrix invertible modulo q.

2.3. **Core Theorem.** For every $a \in \mathbb{Z}_q^{kn}$ the proof of Theorem 1 below provides with an algorithm to compute $y \in \mathbb{Z}_q^n$ and $z \in \mathbb{Z}^{kn}$ with $\max(z) \leq c$ such that $a \equiv Ty + z \mod q$. Since the matrix T is triangular, the algorithm repeatedly solves the system (1) for different $b_1, \dots, b_k$ to construct y and z. So (1) should admit an efficient solution for every diagonal tuple $[t_{1j}, t_{2j}, \dots, t_{kj}]$ in the definition of T.

One can use $k = 2, q \leq (2c+1)^2$ and $[t_1, t_2] = [1, 2c+1]$ since an efficient algorithm to solve (1) is provided by Section 1.3.2 in that case. Theorem 1 is applied to generate signatures in Section 2.4.2.

Theorem 1. *For every* $a \in \mathbb{Z}_q^{kn}$*, there exist* $y \in \mathbb{Z}_q^n$ *and* $z \in \mathbb{Z}^{kn}$ *such that* $\max(z) \leq c$ *and* $a \equiv Ty + z \mod q$*.*

We denote $a = (a_1, a_2, \dots, a_{kn})$, $y = (y_1, y_2, \dots, y_n)$, and $z = (z_1, z_2, \dots, z_{kn})$. Let T_j be a sub-matrix of T of size $k \times j$ in the rows $jk+1, jk+2, \dots, jk+k$ and columns $1, \dots, j$, where $0 \leq j \leq n-1$. The entries of T_j are denoted by $*$ in the definition of T. The following algorithm computes $y \in \mathbb{Z}_q^n$ and $z \in \mathbb{Z}^{kn}$ specified in the theorem and thus proves it.

(1) For $0 \leq j \leq n-1$ set

$$\begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_k \end{pmatrix} \equiv \begin{pmatrix} a_{jk+1} \\ a_{jk+2} \\ \dots \\ a_{jk+k} \end{pmatrix} - T_j \begin{pmatrix} y_1 \\ \dots \\ y_j \end{pmatrix} \mod q,$$

and $[t_1, \dots, t_k] = [t_{1\,j+1}, t_{2\,j+1}, \dots, t_{k\,j+1}]$,

(2) and compute any solution u to the system (1). Set $y_{j+1} = u$ and

$$\begin{pmatrix} z_{jk+1} \\ z_{jk+2} \\ \dots \\ z_{jk+k} \end{pmatrix} \equiv \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_k \end{pmatrix} - \begin{pmatrix} t_{1\,j+1} \\ t_{2\,j+1} \\ \dots \\ t_{k\,j+1} \end{pmatrix} y_{j+1} \mod q$$

Therefore for every $0 \leq j \leq n-1$ we have

$$\begin{pmatrix} a_{jk+1} \\ a_{jk+2} \\ \dots \\ a_{jk+k} \end{pmatrix} \equiv \begin{pmatrix} & t_{1\,j+1} \\ & t_{2\,j+1} \\ T_j & \dots \\ & t_{k\,j+1} \end{pmatrix} \begin{pmatrix} y_1 \\ \dots \\ y_j \\ y_{j+1} \end{pmatrix} + \begin{pmatrix} z_{jk+1} \\ z_{jk+2} \\ \dots \\ z_{jk+k} \end{pmatrix} \mod q.$$

So $a \equiv Ty + z \mod q$, where $|z_i| \leq c$. That proves the theorem.

2.4. **Algorithms.**

In this section basic EHTv3 routines are presented. The signature scheme consists of private and public key generating algorithms, signature generating and signature verifying algorithms.

2.4.1. *Public and Private Keys Generation.*

(1) Randomly generate $C = (C_1 | C_2) \in \mathbb{Z}^{m \times kn}$ according to Section 2.2.2 and Section 6 specifications. If C_1 is not invertible modulo q, then repeat. Otherwise, continue.
(2) Choose diagonal tuples $[t_{1j}, t_{2j}, \dots, t_{kj}]$ for the matrix $T \in \mathbb{Z}_q^{kn \times n}$, see Section 1.3.1. For $k = 2$ and $q \leq (2c+1)^2$ one may set every diagonal tuple to $[1, 2c+1]$. Randomly generate the entries of T under the diagonal entries.
(3) Randomly generate $B \in \mathbb{Z}_q^{n \times n}$. If B is invertible, then construct B^{-1} and continue. Otherwise, repeat.
(4) Set $A = CTB^{-1} \in \mathbb{Z}_q^{m \times n}$.

The private key may be produced from a short seed. Take a random seed, generate C, T, B by clocking a pseudo random number generator (PRNG) initialised by this seed. If both C_1 and B are invertible, then compute A and keep the seed. Otherwise, take a new seed and repeat.

2.4.2. *Signature Generation.*

(1) Let M be a message. Compute $h = \mathrm{HASH}(M) \in \mathbb{Z}_q^m$.
(2) Generate private key matrices C, T, B from the seed.
(3) Randomly generate $a'' \in \mathbb{Z}_q^{kn-m}$. Compute $a' \in \mathbb{Z}_q^m$ by solving the system of linear equations $C_1 a' \equiv h - C_2 a'' \mod q$. Set $a = (a'|a'') \in \mathbb{Z}_q^{kn}$.
(4) Compute $y \in \mathbb{Z}_q^n$ and $z \in \mathbb{Z}^{kn}$ such that $a \equiv Ty + z$ and $\max(z) \leq c$ by Theorem 1, Section 2.3.
(5) Compute $e = Cz$. If $\max_l(e) \leq s$, then the signature for M is $x \equiv By \in \mathbb{Z}_q^n$, otherwise repeat with a new a''.

2.4.3. *Signature Verification.*

(1) To verify the signature $x \in \mathbb{Z}_q^n$ for M, compute $h = \mathrm{HASH}(M) \in \mathbb{Z}_q^m$.
(2) Compute $e \equiv h - Ax \in \mathbb{Z}_q^m$ and take the entries of e smallest in absolute value residues modulo q.
(3) If $\max_l(e) \leq s$, then accept the signature, otherwise reject.

2.5. **Verification Proof.** We have

$$Ca \equiv h, \qquad a \equiv Ty + z \mod q,$$

where $z \in \mathbb{Z}^{kn}$, and $\max(z) \leq c$, and $x \equiv By$. Then

$$a \equiv Ty + z \equiv TB^{-1}x + z \mod q.$$

So $h \equiv Ax + e \mod q$, where $e \equiv Cz \mod q$. One takes the entries of $e \in \mathbb{Z}_q^m$ smallest in absolute value residues modulo q. As $s < (q-1)/2$, we get $\max_l(e) \leq s$. The signature is accepted.

3. **Signature distribution**

In this section we show that the signature x is close to be uniformly distributed on $\mathbb{Z}_q^n$ if the hash function provides a uniform distribution on $\mathbb{Z}_q^m$.

Theorem 2. *Let a be distributed uniformly on $\mathbb{Z}_q^{kn}$ and let $y \in \mathbb{Z}_q^n$ and $z \in \mathbb{Z}^{kn}$, where $a \equiv Ty + z \mod q$ and $\max(z) \leq c$, be produced with Theorem 1. Then y is uniformly distributed on $\mathbb{Z}_q^n$.*

Proof. Since a is distributed uniformly on $\mathbb{Z}_q^{kn}$, each entry of y is produced by solving (1) for diagonal tuples $[t_{1j}, t_{2j}, \ldots, t_{kj}]$ and for independent uniformly distributed $b_1, \ldots, b_k$. One can there put $t_1 = t_{1j} = 1, t_2 = t_{2j} \ldots, t_k = t_{kj}$ to simplify the notation below.

So (1) is equivalent to the following statement. For any residues $b_1, \ldots, b_k$ modulo q there exist a residue u and integers $z_1, \ldots, z_k$, where $|z_1| \leq c, \ldots, |z_k| \leq c$ and

$$b_1 \equiv u + z_1,\ b_2 \equiv ut_2 + z_2, \ldots,\ b_k \equiv ut_k + z_k \mod q. \tag{3}$$

Let $A(b_1, \ldots, b_k)$ denote the set of such u. It is enough to prove that if $b_1, \ldots, b_k$ are generated independently and uniformly at random on residues modulo q and the solution u to (1) is taken uniformly from $A(b_1,\ldots,b_k)$, then u is uniformly distributed on residues modulo q. The probability of u is obviously equal to

$$\frac{1}{q^k}\sum_{u\in A(b_1,\ldots,b_k)}\frac{1}{|A(b_1,\ldots,b_k)|},$$

where the sum runs over all $b_1,\ldots,b_k$ such that $u\in A(b_1,\ldots,b_k)$. The following lemma implies that this probability is $1/q$.

Lemma 1. $\sum_{u\in A(b_1,\ldots,b_k)}\frac{1}{|A(b_1,\ldots,b_k)|}=q^{k-1}$.

The system (3) is equivalent to

$$b_1-\ u\equiv z_1\,, b_2-t_2b_1\equiv z_2-t_2z_1\,,\ldots,b_k-t_kb_1\equiv z_k-t_kz_1,$$

where $|z_1|\le c,\ldots,|z_k|\le c$. For any $a_1,\ldots,a_k$ let $s=s(a_2,\ldots,a_k)$ denote the number of solutions $z=(z_1,\ldots,z_k)$ to

(4) $$|z_1|\le c,\ldots,|z_k|\le c,\qquad a_2\equiv z_2-t_2z_1,\ldots,a_k\equiv z_k-t_kz_1.$$

Then

$$|A(b_1,\ldots,b_k)|=s(a_2,\ldots,a_k),$$

where $a_2\equiv b_2-t_2b_1,\ldots,a_k\equiv b_k-t_kb_1$. Moreover, $u\in A(b_1,\ldots,b_k)$ if and only if $b_1=u+z_1, b_2=a_2+t_1b_1,\ldots,b_k=a_k+t_kb_1$, where $z_1,\ldots,z_k$ is a solution to (4) for any $a_2,\ldots,a_k$. Since $a_2,\ldots,a_k$ may take any values, we get

$$\sum_{u\in A(b_1,\ldots,b_k)}\frac{1}{|A(b_1,\ldots,b_k)|}=\sum_{a_2,\ldots,a_k}\ \sum_{z_1,\ldots,z_k}\frac{1}{s(a_2,\ldots,a_k)}=\sum_{a_2,\ldots,a_k}1=q^{k-1},$$

where the internal sum is over all the solutions $z=(z_1,\ldots,z_k)$ to (4). The lemma and the theorem are proved. □

The theorem implies that if the probability of $\max_l(e)\le s$ is close to 1, then the distribution of the signature $x\equiv By \mod q$ is close to the uniform on $\mathbb{Z}_q^n$.

For uniformly random $b_1,\ldots,b_k$ modulo q we now study the distribution of $z=(z_1,\ldots,z_k)$, where $|z_1|\le c,\ldots,|z_k|\le c$ and (3) holds. It is easy to see that the probability of z is

$$\frac{1}{q^{k-1}s(a_2,\ldots,a_k)},$$

where $a_2\equiv z_2-t_2z_1,\ldots,a_k\equiv z_k-t_kz_1 \bmod q$. So, the distribution of z may be slightly non-uniform.

Let, for instance, $q=9,k=3,c=2$. There is only one tuple $[t_1,t_2,t_3]=[1,2,4]$ up to the equivalence for these parameters. We now compute the mean and the variance of $|z|^2=z_1^2+\ldots+z_k^2$ in this case. By direct calculation, the mean of $|z|^2$ is $16/3\approx 5.33$ and its variance is $236/27\approx 8.74$. If $z_1,\ldots,z_k$ are distributed independently and uniformly over $-c,\ldots,c$, then the mean of $|z|^2$ is 6 and the variance is 8.4, which do not differ significantly. The distribution of z may be uniform too as for $q=25,k=2,c=2$.

4. **EHTv3 Efficiency**

The matrices C_1 and B must be invertible modulo q. This happens with a significant probability. If not, then the matrices are discarded and generated again. One can generate B as a product of lower and upper triangular matrices invertible modulo q as in LU decompositions, see [9]. That also makes public key generation more efficient because triangular matrices are easy to invert.

We summarise the efficiency of the signature algorithm. Public key size, private key (seed) size and signature size are measured in bits in Table 2. Public key matrix A may be encoded by $\lceil mn \log_2 q \rceil$ bits. For more efficient operations the matrix may be encoded entry-wise, that is by using $mn\lceil \log_2 q \rceil$ bits. For security levels in Section 6 it is enough to take a 384-bit (48-byte) seed. The complexity of

Table 2. Storage

storage parameters	bits
public key	$\lceil mn \log_2 q \rceil$ or $mn\lceil \log_2 q \rceil$
private key: seed and char. polynomial	$384 + \lceil m \log_2 q \rceil$
signature	$\lceil n \log_2 q \rceil$

private key generation is shown, see Table 3, in the number of the pseudo-random number generator clocks according to Section 4. The complexity of public key generation, signature generation and signature verification is there shown in the number of operations modulo q. In signature generation one must solve $Ca \equiv h \mod q$ for a given h. This is reduced to solving a system of linear equations with the sparse matrix C_1 of size $m \times m$ with τ non-zero entries ± 1 in each row. One may use Cayley-Hamilton theorem to this end and therefore one needs the characteristic polynomial for C_1. The cost of computing the polynomial is at most m^3 multiplications modulo q. Though the matrix C_1 is secret, its characteristic polynomial may not be kept secret. Since each row of C_1 is of 1-norm τ, the cost of the Cayley-Hamilton theorem application is τm^2 additions/subtractions modulo q. The polynomial is a part of the private key, though not necessarily secret. Another option is to use Berlekamp-Massey algorithm as in [21] to construct an appropriate polynomial for solving each particular $Ca \equiv h \mod q$. No need to keep the characteristic polynomial in that case.

Table 3. Time Complexity

algorithm	PRNG clocks/operations mod q
matrix T	$kn(n-1)\lceil \log_2 q \rceil/2$ clocks
matrix B	$n^2\lceil \log_2 q \rceil$ clocks
matrix C	$\lambda m\lceil \log_2 kn \rceil + \lambda m$ clocks
char. polynomial for C_1	m^3 op. mod q
public key matrix A	$kn^3 + \lambda mn$ op. mod q
signature generation	$m(\lambda - \tau) + \tau m^2 + kn(n-1)/2 + n^2$ op. mod q
signature verification	mn op. mod q

Some calculations may repeat few times in order to get an invertible C_1 and a valid signature, see Section 2.4. Asymptotically, the time complexity of the public key generation is $O(n^3)$, the time complexities of the private key generation, signature generation and signature verification are $O(n^2)$. The public key size is $O(n^2)$. The signature size and the private key size are $O(n)$.

5. Cryptanalysis

There are three approaches to the cryptanalysis of EHTv3: find private key given public key only, find private key by analysing a number of valid signatures, and forge signatures without knowledge of the private key.

5.1. **Private Key Recovery and Algebraic Attacks.** One can reduce the private key recovery problem to solving a system of equations modulo q. To simplify the notation, we assume that $k = 2$. Let $B_1, \ldots, B_n$ be unknown columns of B, and let $C_1, \ldots, C_{2n}$ be unknown columns of C. Also, let $T_i = (0, \ldots, 0, 1, t, x_{2i+1}, \ldots, x_{2n})$ be the i-th column of T for unknown residues x_i modulo q. As $CT = AB$, it holds that

$$C_{2i-1} + tC_{2i} + x_{2i+1}C_{2i+1} + \ldots + x_{2n}C_{2n} \equiv AB_i \mod q. \tag{5}$$

This generally results in a system of $2n$ quadratic equations. For $i = n$ the equation (5) transforms into the system $C_{2n-1} + tC_{2n} = AB_n$ of m linear equations, the variables of which are the entries of B_n, C_{2n-1}, C_{2n}, overall $n + 2m$ unknowns, so the number of solutions is around q^{n+m}, which is too big.

One may guess the last two columns of C, solve the linear system for B_n. Each of the last $d = 2n - m$ columns of C contains around $m(\lambda - \tau)/d$ entries ± 1 on the average and the rest entries are zeroes. So the probability of guessing is very low for the parameters in Section 6.

Alternatively, one may guess the positions of n zero entries of $V = C_{2n-1} + tC_{2n}$, solve a system of n homogeneous linear equations in n variables, the entries of B_n, check the guess and finally recover V and therefore C_{2n-1}, C_{2n}. When constructing the matrix C, the columns C_{2n-1}, C_{2n} may be chosen to have non-zero entries in distinct positions. So the vector V contains around $2m(\lambda - \tau)/d$ non-zero entries and therefore around θm zeros, where $\theta = 1 - 2(\lambda - \tau)/d$. The success probability is then

$$P_{Al} = \frac{\binom{\theta m}{n}}{\binom{m}{n}}.$$

For EHTv3 NIST category 1 the parameters are $q = 47, c = 3, \lambda = 9, \tau = 4, m = 460, l = 451, s = 13, d = 24$, so $n = 242$. The success probability $P_{Al} = 2^{-275.03}$. This method is not efficient even if run on a quantum computer. For similar reasons, these methods do not seem efficient for solving (5) if $i < n$.

5.2. **Existential Forgery by Guessing.** Given $h = \mathrm{HASH}(M)$, one may guess small values ($\leq s$ in absolute value) of some n entries of $e \equiv h - Ax \bmod q$, compute x by solving a system of n linear equations modulo q. One then checks if among other $m - n$ entries of e there are at least $l - n$ entries which are at most s in absolute value. If yes, then $\max_l(e) \leq s$ and x passes the verification. Let $p = \frac{2s+1}{q}$ be the probability that a random residue modulo q is at most s in absolute value. The attack success probability is

$$P_G = \sum_{i=l-n}^{m-n} \binom{m-n}{i} p^i \, (1-p)^{m-n-i}.$$

For each guess the cost may be optimised to around $(\log_2 P_G^{-1})(m-n)/2$ additions and subtractions modulo q.

So, on the average, the attack cost is $Q = P_G^{-1}\,(\log_2 P_G^{-1})(m-n)/2$ operations modulo q. For EHTv3 NIST category 1 the parameters are $q = 47, c = 3, \lambda = 9, \tau = 4, m = 460, l = 451, s = 13, d = 24$ so $n = 242$. One computes $Q = 2^{140.69}$ operations modulo q, that provides with $> 2^{143}$ classical gates. The success probability is $P_G = 2^{-126.94}$. That is only slightly higher than the success probability of the key search for AES-128. Therefore the complexities of quantum Grover's search algorithm are approximately the same in the both cases, that is 2^{83} quantum gates according to [10].

5.3. **Existential Forgery by Solving CVP.** Let $h = \mathrm{HASH}(M)$ and let A_I be a sub-matrix of A whose rows are the rows of A with indices $I = \{i_1, \ldots, i_l\}$. Denote by h_I a sub-vector of h whose entries are the entries of h with indices I.

To forge a signature for M one may try to find a vector $e_I \in \mathbb{Z}^l$ with entries bounded by s and $x \in \mathbb{Z}_q^n$ such that $h_I \equiv A_I x + e_I$. Then x verifies.

Let L be a lattice of rank l generated by the columns of A_I modulo q. We can assume that the volume of L is q^{l-n}. In order to forge a signature for M it is enough to solve an approximate CVP for L in the max norm. The solution of this problem implies a solution of CVP for Euclidean norm with a small approximation factor $O(s/q^{1-n/l})$.

One may also apply an exact CVP algorithm for the Euclidean norm as in [3] or [12]. According to [12], that CVP may be solved in heuristic time $2^{0.292\,l+o(l)}$ by sieving with the same amount of memory. However, this attack seems inferior compared with sieving to solve a relevant instance of SVP in the next section.

5.4. **Existential Forgery by Lattice reduction and Sieving.**

5.4.1. *Rank m Lattice Reduction.* Let $\big(h|A\big)$ be a matrix of size $m \times (1+n)$, a concatenation of h and the matrix A. The equation $h \equiv Ax + e \mod q$ may be written as

$$\big(h|A\big)\begin{pmatrix}1\\-x\end{pmatrix} \equiv e \mod q.$$

The vector $e = (e_1, e_2, \ldots, e_m) = Cz$ belongs to a lattice L of rank m and of volume q^{m-n-1} generated by the columns of $\big(h|A\big)$. Since the Euclidean norm $|e|$ is relatively small, one may apply a lattice reduction algorithm to construct x thus forging a signature for h. The best general purpose algorithm is BKZ (Block-Korkin-Zolotarev) introduced in [17]. It is based on solving a Core-SVP problem of finding shortest non-zero vectors in projected sub-lattices of a smaller rank b. The latter depends on the norm of the target vector e besides the parameters of the lattice. According to [1], a smaller $|e|$ results in a smaller b and BKZ works faster. The Core-SVP may be solved with a sieving algorithm in time $2^{0.292\,b+o(b)}$, see [4].

By the signature algorithm, the lattice L contains q^d vectors $e = Cz$. We will study the probability that at least one of them is extremely short, in particular, its norm is around the expected norm of the shortest non-zero vector in a lattice of rank m and volume q^{m-n-1}.

Since $e = Cz$, one may assume that $e_i = z_1 + \ldots + z_\lambda$, where z_i are taken independently and uniformly at random from $[-c, \ldots, c]$ and e_i are distributed independently. There is some slight dependency introduced by the structure of the matrix C which is neglected in the analysis below. The probability that for at least one e it holds that $|e|^2 = e_1^2 + e_2^2 + \ldots + e_m^2 \leq \delta$ is

$$1 - \left[1 - \mathbf{Pr}(|e|^2 \leq \delta)\right]^{q^d} \approx q^d \, \mathbf{Pr}(|e|^2 \leq \delta)$$

for a small enough δ. The probability $\mathbf{Pr}(|e|^2 \leq \delta)$ drops abruptly when δ decreases. For EHTv3 NIST category 1 the parameters are $q = 47, c = 3, \lambda = 9, \tau = 3, m = 460, l = 451, s = 13, d = 24$, so $n = 242$. The expectation of $|e|^2$ is 16560 and its variance is $1.109 \cdot 10^6$. It holds that the shortest non-zero vector $v \in L$ satisfies $|v|^2 \leq 1780$ according to the asymptotical bound for Hermite's constant, see Chapter 2 in [16].

On the other hand, it is easy to estimate $\mathbf{Pr}(|e|^2 \leq 1780) < 3.129 \cdot 10^{-136}$. The number of $e = Cz \in L$ is $q^d = 1.35 \cdot 10^{40}$. The data suggests that the waiting time to get h or/and A with some $e = Cz$ being a shortest non-zero vector in L is in general significantly larger than the advantage of solving the Core-SVP with BKZ.

5.4.2. *Rank l Lattice Sieving.* Let $I = \{i_1, i_2, \ldots, i_l\}$ be a subset of $\{1, 2, \ldots, m\}$ and let A_I be a sub-matrix of A constructed with the rows of A indexed by I. The equation $h \equiv Ax + e$ implies $h_I \equiv A_I x + e_I$, where h_I, e_I are relevant subvectors of h and e respectively. The vector e_I belongs to a lattice L_I generated by the columns of the matrix $\left(h_I | A_I\right)$ modulo q as

$$\left(h_I | A_I\right) \begin{pmatrix} 1 \\ -x \end{pmatrix} \equiv e_I \mod q.$$

Therefore, a short enough vector $e \equiv \left(h_I | A_I\right) y$ in L_I may have all entries bounded by s in absolute values. If, in addition, $y = \begin{pmatrix} 1 \\ -u \end{pmatrix}$, then $x = u$ verifies. The latter happens with probability $2/q$.

One may try to construct a shortest non-zero vector in L_I. By Gaussian Heuristic, its expected Euclidean norm is $\sqrt{l/2\pi e}\; q^{(l-n-1)/l}$. For EHTv3 NIST category 1 parameters ($q = 47, n = 242, l = 451, s = 13$) that equals 30.34. To construct a shortest non-zero vector in L_I, one applies the sieving algorithm in a projected sub-lattice of rank $l' = l - \lfloor l \ln(4/3)/\ln(l/2\pi e) \rceil$ according to [6]. By an argument similar to that in [7], the cost of the sieving is at least $2^{23+0.292\, l'}$ gates. For $l = 451$ the number of gates is at least 2^{143}, the cost of breaking AES-128 on a classical computer.

Let $r = 30.34\, \alpha$ for integers α. We want to evaluate for which α the probability that a random vector of integers of length l and of the Euclidean norm $\leq r$ has all its entries bounded by s. In Table 4, for $1 \leq \alpha \leq 5$, we estimate the number S_r and U_r of the solutions to $x_1^2 + x_2^2 + \ldots + x_l^2 = \lfloor r^2 \rfloor$, where the integers x_i are bounded by s, that is the max norm of $(x_1, \ldots, x_l)$ is bounded by s, and not bounded (bounded by $\lfloor r \rfloor$) respectively, that is the Euclidean norm of $(x_1, \ldots, x_l)$ is bounded by r. We use Cauchy's inequality [20] for the coefficients of the power series $(1 + 2\sum_{i=1}^{t} x^{i^2})^l$ for $t = s$ and $t = \lfloor r \rfloor$ to get upper bounds for S_r and U_r.

The table data suggests that a vector in L_I with the Euclidean norm ≤ 91.02 may have all entries bounded by s, that is its max norm is at most s, with probability close to 1. Hence, to forge a signature for EHTv3 NIST category 1 parameters it may be enough to solve an approximate SVP in the Euclidean norm for the lattice L_I with an approximation factor $\alpha \leq 3$. Since the approximation factor is small, that might be as hard as finding a shortest non-zero vector in L_I. However, that

Table 4. Estimates of S_r and U_r

r	$S_r \le$	$U_r \le$
30.34	$5.49\,10^{347}$	$5.49\,10^{347}$
60.68	$3.61\,10^{483}$	$3.61\,10^{483}$
91.02	$4.38\,10^{562}$	$9.34\,10^{562}$
121.36	$5.14\,10^{614}$	$2.14\,10^{619}$
151.70	$2.02\,10^{642}$	$1.15\,10^{663}$

needs further study as the conclusions are based on the upper bounds for S_r and U_r and not their exact values. A similar analysis holds for EHTv3 NIST category 3 and EHTv3 NIST category 5 parameters.

According to [11], the quantum cost of a sieving based attack is $2^{0.262 \cdot l}$ bit operations. So for EHTv3 NIST category 1 the quantum cost is at least 2^{118} gates. That is larger than breaking AES-128 with Grover's algorithm. Similar is true for other parameters.

5.5. Adaptive Forgery under Known (Chosen) Message Attack.

A message M may have several valid signatures $x_1, x_2, \ldots$. So, we may suppose that the equations

$$h \equiv Ax_1 + e_1, \quad h \equiv Ax_2 + e_2, \ldots$$

are available, where $h = \mathrm{HASH}(M)$. Let a signature x_0 for maybe another message M_0 be available too. Hence, $h_0 \equiv Ax_0 + e_0$, where $h_0 = \mathrm{HASH}(M_0)$. One may try to modify x_0 to yet another valid signature $x_0 + x_1 - x_2$ for M_0 and get

$$h_0 \equiv A(x_0 + x_1 - x_2) + e_0 + e_1 - e_2.$$

The probability to accept $x_0+x_1-x_2$ equals the probability that $\max_l(e_0+e_1-e_2) \le s$. Let p denote the probability that one entry of $e_0 + e_1 - e_2$ is bounded by s. We may assume that the entries of $e_0+e_1-e_2$ are independently distributed. Therefore the probability that $\max_l(e_0 + e_1 - e_2) \le s$ is

$$P_A = \sum_{i=l}^{m} \binom{m}{i} p^i \, (1-p)^{m-i}.$$

For EHTv3 NIST category 1 the parameter are $q = 47, k = 2, c = 3, \lambda = 9, s = 13$. So $p = 0.805$. Since $m = 460, l = 451$, we have $P_A = 2^{-101.14}$. To make the probability of the attack close to 1, one needs a bit more than $2^{101.14}$ independently generated triplets e_0, e_1, e_2. If the triplets are taken from a smaller set, then they are generally dependent and the probability of the success may drop significantly. According to the NIST call, only up to 2^{64} valid signatures are available for the analysis. The attack does not seem work. However, that needs further study, in particular, to learn the attack success probability if the triplets are taken from a set of size 2^{64}. In a variation of this attack $e_0 = e_1$; the probability of $\max_l(2e_1-e_2) \le s$ is even smaller.

5.6. Known Message Attack and HPP.

Suppose a number of valid signatures M_i, x_i were generated on the same public key A and they are available. One can compute $h_i - Ax_i \equiv e_i \mod q$, where $h_i = \mathrm{HASH}(M_i)$ and $e_i = Cz_i$ for $z_i \in \mathbb{Z}^{kn}$, and the entries of z_i are taken from $[-c, \ldots, c]$. The matrix C is rectangular and of size $m \times kn$. The cryptanalyst observes $e_i = Cz_i$ and may learn the Gram matrix

Table 5. Complexity of the attacks for EHTv3 NIST category 1

attack	
guessing, Section 5.2	$2^{140.69}$
sieving ($l' = 411$), Section 5.4	$2^{120.2}$
appr. SVR factor α ($l = 451$), Section 5.4	≤ 3
algebraic, Section 5.1	$2^{275.03}$
forgery probability P_A, Section 5.5	$2^{-101.14}$

$CC^T \in \mathbb{Z}^{m\times m}$. However, reconstructing the matrix C itself seems a hard problem. The method of learning C by solving Hidden Parallelepiped Problem (HPP) [8] is not generally applicable for rectangular matrices. In this approach one minimises the 4-th moment of aCz for a variable vector a since the values of the random variable $e = Cz$ are known. If the matrix C is a row vector (which is a rectangular matrix), then the solution to the minimisation problem is $a = 0$ regardless of C.

6. EHTv3 Proposed Parameters

In this section we propose parameter sets for three security levels. For them we take $q = 47, k = 2, c = 3$. There is only one tuple $[1, 7]$, up to the equivalence, which satisfy the condition in Section 2.2 for these values. Section 1.3.2 provides with an efficient algorithm to solve (1) in that case.

6.1. **NIST security category 1.**

That requires around 2^{128} operations and at least 2^{143} gates to break the system, which is the complexity of breaking AES-128 on a classical computer. We set $m = 460, l = 451, n = 242$ and $\lambda = 9, \tau = 4, s = 13, d = kn - m = 24$. The signature size is 169 bytes, and the public key size is 77.3 Kbytes, and the private key size is 368 bytes (48 bytes of a secret seed and the rest is the non-secret characteristic polynomial of C_1). The probability that $\max_l(e) \leq s$, where $e = Cz$, is around 0.417. This value was computed under an assumption on the matrix C. Experimentally, the average number of trials (the cost of one trial is less than the cost of generating the signature) before generating a valid signature is around 2.6.

The parameters are optimised to balance the complexity of so far best attacks as the guessing algorithm in Section 5.2, the lattice sieving algorithm in Section 5.3 and algebraic attacks in Section 5.1, see Table 5. Also, the parameters are chosen to minimise the size of the signature and the size of the public key. The cost of guessing is presented in the number of additions modulo $q = 47$, the cost of sieving is to be multiplied by at least 2^{23} according to Section 5.4.2, the cost of the algebraic attack is in the number of systems on n linear equations in n variables to solve modulo q, see Table 5. As specified in Section 5.4.2, to forge a signature it may be enough to solve an approximate SVP for the Euclidean norm in a lattice of rank $l = 451$ with the approximation factor 3.

According to Section 5, the number of quantum gates to break EHTv3 NIST category 1 is at least comparable to the complexity of breaking AES-128 with Grover's algorithm.

6.2. **NIST security category 3.**

This category requires around 2^{192} operations and at least 2^{207} gates to break the system, which is the complexity of breaking AES-192 on a classical computer (NIST[14] security category 3). We set $m =$

Table 6. Complexity of the attacks for EHTv3 NIST category 3

attack	
guessing, Section 5.2	$2^{211.99}$
sieving ($l' = 630$), Section 5.4	$2^{184.20}$
appr. SVR factor α ($l = 684$), Section 5.4	≤ 3
algebraic, Section 5.1	$2^{252.39}$
forgery probability P_A, Section 5.5	$2^{-157.79}$

$696, l = 684, n = 367$ and $\lambda = 9, \tau = 4, s = 13, d = kn - m = 38$. The signature size is 255 bytes, and the public key size is 177.36 Kbytes, and the private key size is 532 bytes (48 bytes of a secret seed and the rest is the non-secret characteristic polynomial of C_1). The probability that $\max_l(e) \leq s$, where $e = Cz$, is around 0.217. Experimentally, the average number of trials (the cost of one trial is less than the cost of generating the signature) before generating a valid signature is around 3.22.

The parameters are optimised to balance the complexity of so far best attacks as the guessing algorithm in Section 5.2, the lattice sieving algorithm in Section 5.3 and algebraic attacks in Section 5.1, see Table 5. Also, the parameters are chosen to minimise the size of the signature and the size of the public key. The cost of guessing is presented in the number of additions modulo $q = 47$, the cost of sieving is to be multiplied by at least 2^{23} according to Section 5.4.2. The cost of the algebraic attack is in the number of systems on n linear equations in n variables to solve modulo q, see Table 6. As specified in Section 5.4.2, to forge a signature it may be enough to solve an approximate SVP for the Euclidean norm in a lattice of rank $l = 684$ with the approximation factor 3.

The success probability of guessing is $2^{-197.00}$ and the success probability of guessing in the algebraic attack is $2^{-252.39}$. The quantum cost of sieving is $2^{0.262\,l} = 2^{179.20}$. So by an argument similar to one in Section 5, the number of quantum gates to break EHTv3 NIST category 3 is larger than the complexity of breaking AES-192 with Grover's algorithm.

6.3. **NIST security category 5.** That requires around 2^{256} operations and at least 2^{272} gates to break the system, which is the complexity of breaking AES-256 on a classical computer. We set $m = 940, l = 921, n = 495$ and $\lambda = 9, \tau = 4, s = 13, d = kn - m = 50$. The signature size is 344 bytes, and the public key size is 323.07 Kbytes, and the private key size is 701 bytes (48 bytes of a secret seed and the rest is the non-secret characteristic polynomial of C_1). The probability that $\max_l(e) \leq s$, where $e = Cz$, is around 0.262. Experimentally, the average number of trials (the cost of one trial is less than the cost of generating the signature) before generating a valid signature is around 2.01.

The parameters are optimised to balance the complexity of the guessing algorithm in Section 5.2, the lattice sieving algorithm in Section 5.3 and algebraic attacks in Section 5.1. Also, the parameters are chosen to minimise the size of the signature and the size of the public key. The cost of guessing is presented in the number of additions modulo $q = 47$, the cost of sieving is to be multiplied by at least 2^{23} according to Section 5.4.2, the cost of the algebraic attack is in the number of systems of $n = 495$ linear equations in n variables to solve modulo q, see Table

TABLE 7. Complexity of the attacks for EHTv3 NIST category 5

attack	
guessing, Section 5.2	$2^{269.95}$
sieving ($l' = 854$), Section 5.4	$2^{249.70}$
appr. SVR factor α ($l = 921$), Section 5.4	≤ 3
algebraic, Section 5.1	$2^{241.12}$
forgery probability P_A, Section 5.5	$2^{-202.12}$

7. As specified in Section 5.4.2, to forge a signature it may be enough to solve an approximate SVP for the Euclidean norm in a lattice of rank $l = 921$ with the approximation factor 3.

By an argument similar to Section 5, the number of quantum gates to break EHTv3 NIST category 5 is at least comparable to the complexity of breaking AES-256 with Grover's algorithm.

7. **Reduced public key with EHTv4**

Using finite group ring arithmetic, instead of the arithmetic of residues modulo q, significantly reduces the size of the public key and the time complexity of its generation with keeping security levels. Any finite group ring over $\mathbb{Z}_q$ may be used for the scheme arithmetic. Such rings have natural representation by matrices with entries in $\mathbb{Z}_q$. The multiplication in group rings may be represented by matrix multiplication which we generally want to be non-commutative, as with ordinary matrices, to complicate the analysis.

7.1. **Group Rings and Matrices.** Basic theory of finite groups may be found in [2]. Let $G = \{\alpha_0 = 1, \alpha_1, \ldots, \alpha_{r-1}\}$ be a finite group of order r and let G_q denote a free module over $\mathbb{Z}_q$, whose basis is the elements of G. The set G_q consists of all formal sums $\alpha = a_0\alpha_0 + a_1\alpha_1 + \ldots + a_{r-1}\alpha_{r-1}$, where $a_i \in \mathbb{Z}_q$. That is a group ring with unity 1, which is the unity of G. One may represents the composition law in G by a table. The ring G_q is commutative if and only if the group G is commutative. Since the group size r may be relatively large, we prefer groups where the composition law is easy to implement without keeping composition tables, as in Sections 7.2 below.

7.2. **Some Finite Groups.**

7.2.1. *Cyclic Groups.* For every $r \geq 1$ there exists a cyclic group $G = \{\alpha_0 = 1, \alpha_1 = \alpha, \alpha_2 = \alpha^2, \ldots, \alpha_{r-1} = \alpha^{r-1}\}$ of order r, where α is the group generator. The group law is defined by $\alpha_i\alpha_j = \alpha_{i+j}$, where the indices are reduced modulo r. This group is commutative. Every finite commutative group is a direct sum of its cyclic subgroups.

7.2.2. *Dihedral Groups.* For every enen $r \geq 6$ there exists a non commutative dihedral group D_r of order r. Let $D_r = \{\alpha_0 = 1, \alpha_1, \ldots, \alpha_{n-1}, \beta_0, \beta_1, \ldots, \beta_{n-1}\}$, where $r = 2n$. The group law in D_r is given by

$$\alpha_i\alpha_j = \alpha_{i+j}, \quad \beta_i\beta_j = \alpha_{i-j}, \quad \alpha_i\beta_j = \beta_{i+j}, \quad \beta_i\alpha_j = \beta_{i-j}, \tag{6}$$

where the indices are reduced modulo n.

Table 8. Projective linear groups size

p	4	5	7	8
$\lvert \mathrm{PGL}(2,p)\rvert$	60	120	336	504
$\lvert \mathrm{PSL}(2,p)\rvert$	60	60	168	504

7.2.3. *Projective Linear Groups.* Let $\mathbb{F}_p$ be a finite field of p elements. Projective General Linear group $\mathrm{PGL}(2,p)$ is the quotient of the group of invertible (2×2)-matrices over $\mathbb{F}_p$ by the normal subgroup of all scalar matrices. The elements may be represented by rational functions $\frac{ax+b}{cx+d}$, where $a, b, c, d \in \mathbb{F}_p$ and $ad - bc \neq 0$, that is linear fractional transforms on the projective line $\mathbb{F}_p \cup \infty$. The group law in $\mathrm{PGL}(2,p)$ is the composition of such functions. The size of the group is $p(p^2-1)$, see Table 8. Projective Special Linear group $\mathrm{PSL}(2,p)$ is a group of transforms $\frac{ax+b}{cx+d}$, where $a, b, c, d \in \mathbb{F}_p$ and $ad - bc = 1$. The size of $\mathrm{PSL}(2,p)$ is $p(p^2-1)/2$ for odd p. That is a simple non commutative group for $p \geq 4$.

Let p be an odd prime. We show how to implement the operation in $\mathrm{PSL}(2,p)$. Group elements may be written as tuples (a, b, c, d) over $\mathbb{F}_p$. Every element of the group belongs to one of $p(p^2-1)/2$ equivalence classes split into 3 groups below.

(1) (a, b, c, d), where a is in $[1, 2, \ldots, (p-1)/2]$, and b, c, d are any such that $ad - bc = 1$,
(2) $(0, b, c, d)$, where b, c are any, d is in $[1, 2, \ldots, (p-1)/2]$ such that $0 \cdot d - bc = 1$,
(3) $(0, b, c, 0)$, where b is in $[1, 2, \ldots, (p-1)/2]$ and c is such that $0 \cdot d - bc = 1$.

To compute the composition of $\frac{a_1x+b_1}{c_1x+d_1}$ and $\frac{a_2x+b_2}{c_2x+d_2}$ in $\mathrm{PSL}(2,p)$ one may compute the product of 2×2 matrices

$$\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \equiv \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mod p,$$

and then reduce (a, b, c, d) to one of the classes with the equivalence relation induced by the transform $(a, b, c, d) \to (-a, -b, -c, -d)$

7.2.4. *Permutation Groups.* Let $S_n, n \geq 1$ denote the group of all permutations on n letters and let G be a subgroup of S_n of order r. Since the composition of permutations is easy to implement one only needs to keep the group elements of G to implement the group law in G. Alternating group $A_n \subseteq S_n$ is of order $n!/2$ and of special interest as for $n \geq 5$ this group is simple and not commutative. However, the size of A_n grows fast with n. For instance, $|A_5| = 60$, $|A_6| = 360$, etc.

$\mathrm{PSL}(2,p)$ has a natural representation by permutations on the projective line $\mathbb{F}_p \cup \infty$, that is on $p+1$ letters. Since $\mathrm{PSL}(2,7)$ is isomorphic to the group of all non-singular 3×3 matrices over $\mathbb{F}_2$, it has a representation by permutations on non-zero vectors of length 3 over $\mathbb{F}_2$, that is on 7 letters.

7.3. **Matrices and Norms.** The addition and the multiplication of matrices over G_q follow ordinary definitions. Unity matrix I is a square matrix whose entries are zeros except 1 on the main diagonal. A square matrix B is invertible over G_q if there is another square matrix B_1 of the same size such that $BB_1 = B_1B = I$.

Every element $\alpha \in G_q$ may be written as $\alpha = a_0\alpha_0 + a_1\alpha_1 + \ldots + a_{r-1}\alpha_{r-1}$, where the coefficients $a_0, \ldots, a_{r-1}$ are smallest in absolute value residues modulo q. We then define the norm of α as $|\alpha| = \sum_{i=0}^{r-1} |a_i|$. Let $b = (\beta_1, \ldots, \beta_n)$ be a vector over G_q, then $|b| = \sum_{i=1}^{n} |\beta_i|$ is the norm of b, and $\max(b)$ denotes the maximum in absolute value of all the coefficients of $\beta_1, \ldots, \beta_n$ written with the basis G. Let s be a positive integer. We say $\max_l(b) \leq s$ if at least l such coefficients are at most s in absolute value.

7.4. **Inequalities.** In order to simplify the formulae below, we set $k = 2$ in (1). Let c be a positive integer and let $[1, t]$ be a diagonal tuple, see Section 1.3.1. So (1) is equivalent to the following statement. For every $b_1, b_2 \in \mathbb{Z}_q$ there exists $u \in \mathbb{Z}_q$ such that $b_1 = u + z_1, b_2 = tu + z_2$ in $\mathbb{Z}_q$ and $|z_1|, |z_2| \leq c$. Equivalently, the equation $b_2 - tb_1 = z_2 - tz_1$ has a solution z_1, z_2 such that $|z_1|, |z_2| \leq c$, then $u = b_1 - z_1$. For $q \leq (2c+1)^2$ one may take $t = 2c + 1$, see Section 1.3.2.

Similar holds true in the ring G_q. For every $\beta_1, \beta_2 \in G_q$ there exists $\nu \in G_q$ such that

$$\beta_1 = \nu + \zeta_1, \quad \beta_2 = t\nu + \zeta_2 \tag{7}$$

and every coefficient of $\zeta_1, \zeta_2 \in G_q$ written with the basis G is bounded by c in absolute value. Indeed, (7) is equivalent to $\beta_2 - t\beta_1 = \zeta_2 - t\zeta_1$ and $\nu = \beta_1 - \zeta_1$. Let

$$\begin{aligned}
\beta_2 - t\beta_1 &= a_0\alpha_0 + a_1\alpha_1 + \ldots + a_{r-1}\alpha_{r-1}, \\
\zeta_1 &= z_{10}\alpha_0 + z_{11}\alpha_1 + \ldots + z_{1r-1}\alpha_{r-1}, \\
\zeta_2 &= z_{20}\alpha_0 + z_{21}\alpha_1 + \ldots + z_{2r-1}\alpha_{r-1}.
\end{aligned}$$

Then (7) is equivalent to

$$a_i = z_{2i} - tz_{1i}, \quad i = 0, \ldots, r-1$$

in $\mathbb{Z}_q$. Therefore $z_{2i}, z_{1i} \in \mathbb{Z}_q$ such that $|z_{2i}|, |z_{1i}| \leq c$ do exist and (7) holds for $\nu = \beta_1 - \zeta_1$.

Since the multiplication of $\alpha = a_0\alpha_0 + a_1\alpha_1 + \ldots + a_{r-1}\alpha_{r-1}$, by $\alpha_i \in G$ results in permuting the coefficients of α, the following version holds. For every $\beta_1, \beta_2 \in G_q$ there exists $\nu \in G$ such that

$$\beta_1 = \nu + \zeta_1, \quad \beta_2 = (t\alpha_i)\nu + \zeta_2$$

and every coefficient of $\zeta_1, \zeta_2 \in G_q$ written with the basis G is bounded by c in absolute value.

Let $k \geq 2$ and let $[t_1, \ldots, t_k]$ be a diagonal tuple, see Section 1.3.1, So (1) is true. Then for every $\beta_i \in G_q, i = 1, \ldots, k$ there exists $\nu \in G_q$ such that

$$\beta_1 = t_1\nu + \zeta_1,\ \beta_2 = t_2\nu + \zeta_2, \ldots,\ \beta_k = t_k\nu + \zeta_k$$

and every coefficient of $\zeta_i \in G_q$ written with the basis G is bounded by c in absolute value.

8. **EHTv4 Signature Algorithm**

EHTv4 Signature Algorithm, introduced in this section, may be represented as an instance of EHTv3 Signature Algorithm with parameters comparable to those in Section 6 for the same security levels. However, since the computations are in the group ring G_q instead of $\mathbb{Z}_q$, EHTv4 is significantly more efficient. In particular, at the expense of a larger signature size, the public key size is several times smaller.

8.1. **Parameters.**
Let n, m, r and q, k, c, λ, s, l be positive integers, where $s < (q-1)/2$. Suppose G is a finite group of order r. Its group ring G_q over $\mathbb{Z}_q$ defines the arithmetic of the scheme. Also, let $h = \mathrm{HASH}(M)$ denote a hash value of the message M, encoded by a vector in G_q^m. As in EHTv3, FIPS202_SHAKE256 was used to hash the message M and to obtain an output of byte length so that we have sufficient bytes for $h \in G_q^m$. With our proposed parameter sets, it is always greater than 64 bytes and ensures 256-bit security for h.

8.2. **Private Key.**
The private key consists of three matrices C, T, B over G_q.

8.2.1. *Matrix T.* The matrix T is a $kn \times n$ matrix as

$$T = \begin{pmatrix} t_{11} & 0 & \dots & 0 \\ t_{21} & 0 & \dots & 0 \\ t_{k1} & 0 & \dots & 0 \\ * & t_{12} & \dots & 0 \\ * & t_{22} & \dots & 0 \\ * & t_{k2} & \dots & 0 \\ * & * & \dots & t_{1n} \\ * & * & \dots & t_{2n} \\ * & * & \dots & t_{kn} \end{pmatrix}, \tag{8}$$

where the entries $t_{1j}, t_{2j}, \dots, t_{kj}$ are called diagonal. The entries of T below and to the left of the diagonal are denoted by $*$, they are secret and may be chosen randomly. Each tuple $[t_1, t_2, \dots, t_k]$ of diagonal entries must have the following properties. At least one t_i is invertible in G_q, and for every $\beta_i \in G_q, i = 1, \dots, k$ there exists $\nu \in G_q$ such that

$$\beta_1 = t_1\nu + \zeta_1,\ \beta_2 = t_2\nu + \zeta_2, \dots,\ \beta_k = t_k\nu + \zeta_k, \tag{9}$$

and every coefficient of $\zeta_i \in G_q$ written with the basis G is bounded by c in absolute value. In order to simplify calculations one may take the diagonal entries from $\mathbb{Z}_q$, they must satisfy (1), or those multiplied by the elements of G, see Section 7.4. For such constructed T, a theorem similar to Theorem 1 holds.

Theorem 3. *For every $a \in G_q^{kn}$, there exist $y \in G_q^n$ and $z \in G_q^{kn}$ such that $\max(z) \le c$ and $a = Ty + z$.*

The proof of this theorem closely follows the proof of Theorem 1. The latter also gives an algorithm to compute such y and z given a.

8.2.2. *Matrix C.* The matrix C is a $(m \times kn)$ -matrix over G_q. The norm of each row of C is λ. The matrix is constructed as a concatenation $C = (C_1|C_2)$, where C_1 is a $(m \times m)$ -matrix invertible over G_q. The matrix C_2 is of size $m \times (kn - m)$.

8.2.3. *Matrix B.* The matrix B is an arbitrary invertible $(n \times n)$ -matrix over G_q.

8.3. **Algorithms.**

8.3.1. *Public and Private Keys Generation.*

(1) Randomly generate $C = (C_1|C_2) \in G_q^{m\times nk}$ according to Section 8.2.2 and Section 10 specifications. Reduce the matrix C_1 of size $m \times m$ to a row echelon form. If C_1 is not invertible, then repeat.
(2) Choose diagonal tuples $[t_{1j}, t_{2j}, \dots, t_{kj}], j = 1, \dots, n$ for the matrix $T \in G_q^{kn\times n}$ to satisfy (9). For $k = 2$ and $q \le (2c+1)^2$ one may set every diagonal tuple to $[t_{1j}, t_{2j}] = [1, 2c+1]$. Randomly generate the entries of T under the diagonal entries. Set all other entries of T to 0.
(3) Randomly generate $B \in G_q^{n\times n}$. Reduce B to a row echelon form. If B is invertible, then construct B^{-1}. Otherwise, repeat. The matrix B may be constructed as a product of upper and lower triangular and permutation matrices, see Section 10.
(4) Set the public key matrix $A = CTB^{-1} \in G_q^{m\times n}$. The matrices C, T, B is the system private key.

The private key may be produced from a short seed. Take a random seed, generate C, T, B. Keep the seed if both C_1 and B are invertible, otherwise take a new seed and repeat.

8.3.2. *Signature Generation.*

(1) Let M be a message. Compute $h = \mathrm{HASH}(M) \in G_q^m$.
(2) Generate private key matrices C, T, B from the seed.
(3) Randomly generate $a'' \in G_q^{kn-m}$. Compute $a' \in G_q^m$ by solving the system of linear equations $C_1 a' = h - C_2 a''$. Set $a = (a'|a'') \in G_q^{kn}$.
(4) Compute $y \in G_q^n$ and $z \in G_q^{kn}$ such that $a = Ty + z$ and $\max(z) \le c$ by Section 7.4 and Theorem 3.
(5) Compute $e = Cz$. If $\max_l(e) \le s$, then the signature for M is $x = By \in G_q^n$. Otherwise repeat with a new a''.

8.3.3. *Signature Verification.*

(1) To verify the signature x for M, compute $h = \mathrm{HASH}(M)$.
(2) Compute $e = h - Ax \in G_q^m$.
(3) Accept the signature if $\max_l(e) \le s$. Otherwise reject.

Every correctly generated signature verifies. The formal proof repeats that in Section 2.4.3.

9. **EHTv4 Efficiency**

In this section we summarise the efficiency of the signature algorithm EHTv4. Public key size, private key (seed) size and signature size are measured in bits in Table 9. For NIST security levels it is enough to take a 384-bit (48-byte) seed.

TABLE 9. Storage

storage parameters	bits
public key	$\lceil mnr \log_2 q \rceil$
private key: seed and inversions	$384 + \lceil mr \log_2 q \rceil$
signature	$\lceil nr\ \log_2 q \rceil$

The complexity of public key generation, signature generation and signature verification is measured in the number of the pseudo-random number generator clocks according to Section 4 and in the number of operations modulo q, see Table 10. Some parts of the algorithm as generating C and y, z may repeat few times and that is neglected in Table 10. For $m = 3, n = 2$ a more precise efficiency analysis is in Section 10.3.

Since B must be invertible, it may be constructed as a product of upper and lower triangular and permutation matrices, see Section 10 below, for instance. The inversion of B is then easy to construct. In signature generation one must solve $Ca = h$ for a given h. This is reduced to solving the system of linear equations

$$C_1 a' = h - C_2 a'' \tag{10}$$

over G_q with the sparse square matrix C_1 of size $m \times m$. There exist several ways to handle the problem. The first two methods only work for commutative group rings G_q. The last one works both for commutative and non commutative cases.

(1) One may use Cayley-Hamilton theorem to solve (10) and therefore one needs the characteristic polynomial for C_1 over G_q. The cost of computing the polynomial is around $m^3r^2 + mr^3$ multiplications modulo q, since one needs m inversions in G_q. Let ω be the norm of C_1, that is the sum of norms of the rows of C_1. The cost of the Cayley-Hamilton theorem application is $m\omega$ additions/subtractions in G_q. The polynomial is a part of the private key, though not necessarily secret.
(2) Another option is to use Berlekamp-Massey algorithm as in [21] to construct a minimal polynomial (or its multiple) of $h - C_2a''$ with respect to C_1 for solving each particular (10).
(3) Alternatively, one may solve (10) for a' by reducing C_1 to a row echelon form each time one constructs the signature for h. One applies row operations to $C_1|C_2|h$ to get $I|C_2'|h'$, where I is an $(m \times m)$ unity matrix. Then $a' = h' - C_2'a''$. One may accelerate the computation by keeping up to m inversions in G_q of intermediate entries of C_1 when reducing that to a row echelon form for generating private and public keys. That takes at most $\lceil mr \log_2 q \rceil$ bits as in Table 9. For each h the computation of $I|C_2'|h'$ is to be done only once. However, to generate a valid signature for h one may have to compute $a' = h' - C_2'a''$ for several randomly taken a''.

TABLE 10. Time Complexity

algorithm	PRNG clocks/operations mod q
matrix T	$rkn(n-1)\lceil \log_2 q \rceil/2$ clocks
matrix B	$rn^2\lceil \log_2 q \rceil$ clocks
matrix C	$\lambda m \lceil \log_2 r \rceil + \lambda m$ clocks
row echelon form for C_1	$r^2m^3/3 + r^3m$ operations mod q
public key matrix A	$\lambda mnr + mn^2r^2$ operations mod q
solving the system $Ca = h$	knm^2r^2 operations mod q
constructing the signature	$(kn(n-1)/2 + n^2)\, r^2$ operations mod q
signature verification	mnr^2 operations mod q

One may choose the parameters m, n, r such that the size of the signature is comparable with that in EHTv3. The public key size is then around r times smaller and constructing public keys is around m^2 times faster for a small enough m.

10. **EHTv4 Proposed Parameters**

In this section we propose parameter sets for NIST security categories 1 and 5.

10.1. **NIST security category 1.** The scheme arithmetic is in the group ring G_q, where $G = \mathrm{PSL}(2,7)$ is a simple group of order $r = 168$ represented as $\mathrm{GL}(3,2)$, see Section 7.2.3. We set $(m,n,r) = (3,2,168)$ and $q = 439, k = 2, c = 10$. The tuple $[1,21]$ satisfies the condition in Section 8.2 for these values. We set $\lambda = 54$, $s = 100$, and $l = 492$. The matrix C is of size 3×4 over G_q, whose entries c_{ij} and c'_{ij} satisfy $|c_{ij}| = 1$ and $|c'_{ij}| = 26$, see (11). Every row of C has norm $\lambda = 54$. Public key matrix is

$$A = \begin{pmatrix} c_{11} & c_{12} & c'_{13} & c'_{14} \\ c_{21} & c'_{22} & c_{23} & c'_{24} \\ c'_{31} & c_{32} & c_{33} & c'_{34} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 21 & 0 \\ * & 1 \\ * & 21 \end{pmatrix} \begin{pmatrix} * & * \\ * & * \end{pmatrix}, \tag{11}$$

where the entries of B^{-1} and the entries of T below and to the left of the diagonal tuples are denoted by $*$.

The size of the signature is 369 bytes and the public key size is 1107 bytes. The private key size is 419 bytes (48 bytes of a secret seed and the rest is essentially the inversions of 2 group ring elements). Experimentally, the average number of trials before getting a valid signature $r_s = 4.97$.

The probability to guess each c'_{ij} is around $2^{-129.88}$. Any instance of the signature algorithm EHTv4 may obviously be transformed into an instance of EHTv3 with the public key matrix of size $mr \times nr$ and the signature of size nr over $\mathbb{Z}_q$. That makes the cryptanalysis specified in Section 5 applicable for EHTv4. Table 11 summarises the time complexity of the attacks. As in Section 5, the complexity of sieving in a lattice of rank $l' = 449$ is to be multiplied by 2^{23} to get an estimate of the number of classical gates. The complexity of the algebraic attack is to be multiplied by the complexity of solving systems of nr linear equations in nr variables in operations modulo q, that is at least by $2^{24.8}$. Since the adaptive forgery probability is $2^{-101.36}$, the number of available signatures 2^{64} generated on the same key does not seem enough, see Section 5.5. The approximation factor α for solving an approximate SVP in a lattice of rank $l = 492$ was estimated with Cauchy's inequality as in Section 5.3. However, for these parameters such estimates are not very reliable as the norm of the target vector is too large compared with the dimension of the lattice. We conjecture it to be around 20.

10.2. **NIST security category 5.** The scheme arithmetic is in the group ring G_q, where $G = A_6$ is the alternating group on 6 letters, this group is simple, of order $r = 360$, see Section 7.2.3. We set $(m,n,r) = (3,2,360)$ and $q = 839, k = 2, c = 14$. The tuple $[1,29]$ satisfies the condition in Section 8.2 for these values. We set $\lambda = 100$, $s = 244$, and $l = 1076$. The matrix C is of size 3×4 over G_q, whose entries c_{ij} and c'_{ij} satisfy $|c_{ij}| = 1$ and $|c'_{ij}| = 49$, see (12). Every row of C has

Table 11. Complexity of the attacks for EHTv4 NIST category 1

attack	Sect.	$(m,n,r)=(3,2,168)$
private key recovery, algebraic	5.1	$> 2^{145.59}$
forgery by guessing	5.2	$2^{139.59}$
SVP by sieving ($l'=449$)	5.4	$2^{131.32}$
appr. SVR factor α ($l=492$)	5.4	20
forgery probability P_A	5.5	$2^{-101.36}$

Table 12. Complexity of the attacks for EHTv4 NIST category 5

attack	Sect.	$(m,n,r)=(3,2,360)$
private key recovery, algebraic	5.1	$> 2^{379.60}$
forgery by guessing	5.2	$2^{268.40}$
SVP by sieving ($l'=1001$)	5.4	$2^{292.77}$
appr. SVR factor α ($l=1076$)	5.4	30
forgery probability P_A	5.5	$2^{-127.11}$

norm $\lambda = 100$. Public key matrix is

$$A = \begin{pmatrix} c_{11} & c_{12} & c'_{13} & c'_{14} \\ c_{21} & c'_{22} & c_{23} & c'_{24} \\ c'_{31} & c_{32} & c_{33} & c'_{34} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 29 & 0 \\ * & 1 \\ * & 29 \end{pmatrix} \begin{pmatrix} * & * \\ * & * \end{pmatrix}, \tag{12}$$

where the entries of B^{-1} and the entries of T below and to the left of the diagonal tuples are denoted by $*$. The size of the signature is 875 bytes and the public key size is 2623 bytes. The private key size is 925 bytes (48 bytes of a secret seed and the rest is essentially the inversions of 2 group ring elements). Experimentally, the average number of trials before getting a valid signature $r_s = 3.46$.

The probability to guess each c'_{ij} is around $2^{-256.63}$. Table 12 summarises the time complexity of the attacks. As in Section 5, the complexity of sieving in a lattice of rank $l' = 1001$ is to be multiplied by 2^{23} to get an estimate of the number of classical gates. The complexity of the algebraic attack is to be multiplied by the complexity of solving systems of nr linear equations in nr variables modulo q, that is at least by $2^{28.4}$. Since the adaptive forgery probability is $2^{-127.11}$, the number of available signatures 2^{64} generated on the same key does not seem enough, see Section 5.5. We were not able to estimate the approximation factor α for solving an approximate SVP in a lattice of rank $l = 1076$ as in Section 5.3 since the Cauchy's inequality does not provide sensible results. We conjecture it to be around 30.

10.3. EHTv4 Efficiency for $m = 3, n = 2$.

An invertible matrix B may be generated as

$$B = \begin{pmatrix} u & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix} \begin{pmatrix} z & 1 \\ 1 & 0 \end{pmatrix}$$

for any $u, x, y, z \in G_q$. So computing B takes 5 multiplications in G_q. The inversion of B may be computed as

$$B^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -z \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -y & 1 \end{pmatrix} \begin{pmatrix} 1 & -x \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -u \end{pmatrix}$$

Table 13. EHTv4 efficiency for $m = 3, n = 2$

	multiplications in G_q	inversions in G_q
Key Generation	$3r_k + 16$	$2r_k$
Signature Generation	$11r_s + 19$	2 or 0
Signature Verification	6	0

with 5 multiplications as well. However, it is not necessary to compute these matrices as it is more efficient to multiply subsequently by the diagonal matrices from the definition of B, B^{-1}. For instance, in signature generation it takes only 4 multiplications in G_q to compute $x = By$ without computing B. Table 13 provides with the number of multiplications/inversions in G_q for EHTv4 with $m = 3, n = 2$. We neglect some small costs as the cost of multiplication by the elements of G_q of norm 1 and 2, the cost of inversion of the elements of norm 1. Also, we neglect the cost of the algorithm in Section 1.3.2 for solving (9).

By r_k we denote the number of trials before an invertible matrix C_1 is generated, for the parameter sets above we have $r_k \approx 1$. By r_s we denote the number of trials before a valid signature is constructed. In signature generation it is not necessary to compute inversions in the group ring provided that 2 inversions were produced during key generations and kept as a part of the private key. Below, we provide with the explanation for Table 13 data. In the key generation, the reduction of C_1 to a row echelon form costs 3 multiplications and 2 inversions in G_q. One does that r_k times to get an invertible C_1. The cost of computing the public matrix A is then 16 multiplications. Overall, $3r_k + 16$ multiplications and $2r_k$ inversions.

In the signature generation, the cost of reducing $C_1|C_2|h$ to $I|C_2'|h'$, see Section 9, is 15 multiplications and 2 inversions in G_q. If these 2 inversions are kept from the key generation phase, then this cost is only 15 multiplications. The cost of computing $a = (a'|a'')$ for a random a'' is 3 multiplications and the cost of solving $a = Ty + z$ for $\max(z) \leq c$ is 2 multiplications. Computing $e = Cz$ costs 6 multiplications. The latter three should repeat r_s times to get $\max_l(e) \leq s$. That results in $11r_s$ multiplications. The cost of computing the signature $x = By$ is then 4 multiplications. Overall, $11r_s + 19$ multiplications.

In the signature verification, the cost of computing Ax is 6 multiplications.

11. Advantages and Limitations of EHTv3 and EHTv4

11.0.1. *Advantages.*

(1) The distinct advantage of EHTv3 and EHTv4 is short signatures generated by the schemes.
(2) The EHTv4 public key is only few times larger than the signature itself. That is a clear advantage of EHTv4.
(3) The private key may be generated from a 48-byte seed to satisfy NIST security levels, one can even take a shorter seed. In EHTv3, for efficiency reasons, one also needs to keep the characteristic polynomial (not necessarily secret) of the sub-matrix C_1 of C. In EHTv4 one may keep the inversion of few group ring elements instead.
(4) The schemes are transparent and easy to understand and implement. Public key is generated with essentially one matrix inversion and two matrix-matrix multiplications. The signature is generated with essentially three

matrix-vector multiplications. The verification is done with one matrix-vector multiplication.

(5) Since matrix multiplication is easy to parallelise, the implementation is parallelisable.

(6) Both the schemes allow flexible choice of the parameters to increase security levels if needed.

(7) In EHTv3 the signature is uniformly distributed given $h = \mathrm{HASH}(M)$ is uniform, and that holds if the check $\max_l(e) \leq s$ is omitted. If not, then the signature distribution may slightly deviate from the uniform.

(8) EHTv3 might perform well on 8-bit platforms as its arithmetic is modulo a relatively small positive integer q. Since EHTv3 private key may be produced from a seed, it seems that modern smart cards have computational resources to implement EHTv3 signature generation algorithm. Similar is true for EHTv4 signatures. This direction is to be further explored.

11.0.2. *Limitations.*

(1) EHTv3 public key is rather large and it is a noticeable limitation if compared with EHTv4 and some other lattice based signature schemes.

References

[1] E. Alkim, L. Ducas, Th. Pöppelmann, and P. Schwabe,*Post-quantum key exchange - a new hope*, ePrint Cryptology Archive, 2015/1092.

[2] M. Aschbacher, *Finite Group Theory, 2nd edition*, Cambridge Studies in Advanced Mathematics, Series Number 10, Cambridge University Press, 2000.

[3] A. Becker, N. Gama, A. Joux, *Solving shortest and closest vector problems: The decomposition approach*, IACR Cryptology ePrint Archive, 2013/685.

[4] A. Becker, L. Ducas, N. Gama, and T. Laarhoven, *New directions in nearest neighbor searching with applications to lattice sieving*, in SODA, pp. 10–24, SIAM, 2016.

[5] A. Budroni, I. Semaev, *New Public-Key Crypto-System EHT*, IACR Cryptology ePrint Archive, 2021/234.

[6] L. Ducas, *Shortest Vector from Lattice Sieving: a Few Dimensions for Free*, in Eurocrypt 2018, Part I, LNCS, vol. 10820, pp. 125-145, Springer, 2018.

[7] P.-A. Fouque et al., *Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU*, Specification v1.2-01/10/2020.

[8] P. Q. Nguyen, O. Regev, *Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures,* Journal of Cryptology, vol. 22(2), pp.139–160.

[9] F.R. Gantmacher,*The Theory of Matrices*, vol. 1, Chelsea Publishing Company, N.Y., 1959.

[10] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, *Implementing Grover oracles quantum key search on AES and LowMC*, EUROCRYPT 2020, Part II, LNCS, vol. 12106, pp. 280–310, Springer,2020.

[11] T. Laarhoven, *Search problems in cryptography,* PhD thesis, Eindhoven University of Technology, 2015.

[12] T. Laarrhoven, *Sieving for closest lattice vectors(with preprocessing)*, arXiv: 1607.04789v1, 16 Jul 2016.

[13] D. Micciancio, and S. Goldwasser , *Complexity of Lattice Problems: A Cryptographic Perspective*, The Kluwer International Series in Engineering and Computer Science, vol. 671, Kluwer Academic Publishers, Boston, MA, 2002.

[14] https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[15] Ph. Q. Nguyen and O. Regev, *Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures*, J, Cryptol. vol. 22 (2009), pp. 139–160.

[16] Ph. Q. Nguyen, B. Vallée, editors, *The LLL Algorithm, Survey and Applications,* Springer,2010.

[17] C.-P. Schorr, M. Euchner,*Lattice basis reduction: Improved practical algorithms and solving subset sum problems*, Math. Program., vol.66, pp.181–199, 1994.
[18] I. Semaev, *New Digital Signature Algorithm EHT*, Cryptology ePrint Atchive, 2022/339.
[19] I. Semaev, *New Digital Signature Algorithm EHTv2*, NISK 2022, 28.11-1.12.2022, Kristiansand, Norway.
[20] E. C. Titchmarsh, *The Theory Of Functions (2nd ed.)*, Oxford University Press, 1939.
[21] D. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. on Inf. Theory, vol. 32 (1986), pp. 54–62.