A Signature Scheme from Full-Distance Syndrome Decoding

Abdelhaliem Babiker
aababiker@iau.edu.sa

College of Engineering, Imam Abdulrahman Bin Faisal University

Abstract. In this paper we propose a new hash-and-sign digital signature scheme whose security against existential forgery under adaptive chosen message attack is based on the hardness of full-distance syndrome decoding. We propose parameter sets for three security levels (128-bits, 192-bits, and 256-bits) based on concrete estimations for hardness of the syndrome decoding problem and estimate the corresponding sizes of the keys and the signature for each level. The scheme has large public and private keys but very small signatures.

Keywords: Digital Signature · Syndrome Decoding Problem · Code-Based Cryptography

1 Introduction

It is important to design new public-key cryptographic algorithms that are quantum-resistant given the current efforts towards standardization of post-quantum public key algorithms [1], especially digital signature algorithms.

In this paper we introduce a new hash-and-sign signature scheme that is based on a well-known problem of the *syndrome decoding* which has been studied for long time and believed to be hard for both classic and quantum computers [1, 8]. There have been many attempts to build a hash-and-sign signature schemes that inherit hardness of the syndrome decoding problem. Typically these schemes try to hash the message into a syndrome that is decodable using some secret error-correction code and then decode that syndrome using the secret decoding algorithm and use the corresponding error pattern to define the signature [6, 13]. However, as indicated by [2], the difficulties of transforming the hash to a decodable syndrome have led to either security concerns or unpractical parameters, if not both. The proposed scheme in this paper is devised such that the verification of the validity of the signature corresponds to syndrome decoding instance whose solution is the valid signature. However, instead of trying to find a syndrome that is decodable using some secret code we use the hash value to generate, via matrix-vector multiplication with public random matrix, a random vector that acts as a random syndrome whose corresponding error vector

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

defines the signature, which is also obtained via matrix-vector multiplication. No error correction code is used at all. But nevertheless the public key consists of a random matrix that plays the same role of the parity-check matrix of a random binary linear code with block length n and dimension k.

The scheme is developed in two phases. First we start with a basic digital signature scheme, which is a complete digital signature scheme on its own with keys generation, signing and verifying algorithms. Thence we develop the basic scheme further into an advanced one by modifying the keys and adding a restriction on the verification of the new signature to make it match a full-distance syndrome decoding instance.

In what follows we give a high-level description of the scheme and its security in order to give the reader the big picture before we delve into the details in Sections 3 and 4. So, for simplicity of the presentation the details are neglected; particularly the relation between the keys.

1.1 Digital Signature Scheme: High Level Description

The Keys The formal definition of the keys is a bit complicated. For the purpose of introduction and in order to give an intuitive perception of security of the scheme we present the keys of scheme as follows. The public key is

$$pk^\star = (H^\star, \mathcal{H}, R, d, \theta),$$

where $H^\star \in \mathbb{F}_2^{(n-k)\times n}$, $R \in \mathbb{F}_2^{(n-k)\times L}$, $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^L$ is a secure hash function, $d = n\mathsf{H}^{-1}(1 - \frac{k}{n})$, where $\mathsf{H}(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function, and θ is a positive integer. The private key $sk^\star$ consists of $sk = U$ where $U \in \mathbb{F}_2^{n\times L}$ which is the private key of the basic scheme, in addition to the tuples $S_i = (s_{i1}, \ldots, s_{i\epsilon})$, where $s_{ik} \in [n]$ and $1 \le i \le \theta$, and the matrices $G \in \mathbb{F}_2^{n\times l}$ and $T \in \mathbb{F}_2^{\theta\times l}$, where $l = \theta 2^\epsilon$.

The keys are devised in Sections 3 and 4.

Signing and Verifying Unlike the generation of the keys, signing operation is simple and the verification is even more simple. Consequently, the scheme has efficient signing and verifying algorithms.
The signature of a message $m \in \{0,1\}^*$ is a pair of vectors

$$(\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n), \boldsymbol{\varsigma} = (\varsigma_1, \ldots, \varsigma_\theta)) \in \mathbb{F}_2^n \times \mathbb{F}_2^\theta.$$

We denote by $\mathrm{wt}(\boldsymbol{\sigma})$ the Hamming weight of the vector $\boldsymbol{\sigma}$, and by G_c and T_c we denote the c^{th} columns of the matrices G and T, respectively, and similarly R_k is the k^{th} column of the matrix R. The signing and verifying operations are as follows.

```
Sign$(m, sk^\star)$
    $\mathbf{h} \leftarrow \mathcal{H}(m)$
    $\mathbf{e} = (e_1, \ldots, e_n) \leftarrow U\mathbf{h}$
    $\mathcal{J} \leftarrow \{c \mid c = (i-1)2^\epsilon + \sum_{k=1}^{\epsilon} e_{s_{ik}} 2^{k-1}, 1 \leq i \leq \theta\}$
    $\boldsymbol{\sigma} \leftarrow \sum_{c \in \mathcal{J}} G_c + \mathbf{e}$
    $\varsigma \leftarrow \sum_{c \in \mathcal{J}} T_c$
    **return** $(m, \boldsymbol{\sigma}, \varsigma)$
```

```
Verify$(pk^\star, m, (\boldsymbol{\sigma}, \varsigma))$
    $\mathbf{h} \leftarrow \mathcal{H}(m), \mathbf{s} \leftarrow R\mathbf{h}$
    $\mathbf{s}^\star \leftarrow \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k$
    **if** $\mathrm{wt}(\boldsymbol{\sigma}) = d$ and $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$
    **then** Accept
    **otherwise** Reject
```

That is all. We see sizes of the keys and the the signatures in Table 1 below.

Table 1. Security Levels, and Sizes of the Keys and the Signatures of the Scheme.

Security Level	128-bits	192-bits	256-bits
Public Key (KB)	252.69	583.80	1051.58
Private Key (KB)	54.02	101.41	161.99
Signature (Bytes)	204	309	415

KB = 1024 Bytes.

1.2 Security of the Scheme

We use the notion of *existential unforgeability under adaptive chosen message attacks* EUF-CMA. In particular, as we see from Algorithm Verify above, we let the verification of the signature corresponds to a solution of an instance of the *syndrome decoding problem*. Hence, if the instance is hard, then EUF-CMA security of the scheme is guaranteed. We show this as follows.
The matrix R is uniform random matrix. Thus, it is easy to see that for every message m with hash vector $\mathbf{h}$, the vector $\mathbf{s} = R\mathbf{h}$ is random and out of the adversary's control. To forge a signature for an arbitrary message m one needs to find a pair of vectors $(\boldsymbol{\sigma}, \varsigma)$ which satisfy the identity $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$ with $\mathrm{wt}(\boldsymbol{\sigma}) = d$, where $\mathbf{s}^\star = \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k$. That is, the legitimate signature represents a solution of an instance of a syndrome decoding problem. Thus, forging a signature in this way actually implies solving an instance syndrome decoding problem in which the matrix $H^\star$ plays the role of a parity check matrix for some random linear code and the vector $\mathbf{s}^\star$ represents the syndrome. Therefore, if it is computationally hard to solve the syndrome decoding instance defined by $H^\star$, $\mathbf{s}^\star$, and $\mathrm{wt}(\boldsymbol{\sigma})$, then for a random message m, it is computationally hard to forge a valid signature

using syndrome decoding problem solving algorithms, which, as we assume, is the only feasible way to forge a signature.

Parameters Sets and Security Levels We choose the parameters n and k such that complexity of the syndrome decoding instance (n, k, d) satisfies the required level of security. To this end, we use estimation of Andre Esser and Emanuele Bellini [8] for hardness of the syndrome decoding problem, which is based on the performance of the best known Information Set Decoding algorithms for solving this problem. We determine the parameter sets and their corresponding security level using Esser-Bellini estimations as reference. The parameters sets are shown in Table 2.

Table 2. Parameters Sets for k-bit security ($k = 128, 192, 256$).

Parameters set	128-bits	192-bits	256-bits
n	1600	2432	3264
k	400	608	816
L	125	190	255
ϵ	5	5	5
θ	25	38	51
τ_i $(\approx)$	13	13	13

Paper Organization In Section 2 we define two basic operators for expanding and shrinking matrices and state their properties which we are going to use in devising the basic signature scheme. Next, we design the basic signature scheme in and prove its correctness Section 3. Then, we develop the basic signature scheme into an advanced one in Section 4 by modifying the keys, randomizing the signature, and adding restriction to verification of the basic signature. We define and discuss security of the final scheme in Section 4. And, in Section 5 we provide the security parameters according to an indicative estimations for hardness of our underlying problem. Based on these parameters sets we give estimations for the sizes of the keys and the signatures for the three security levels as shown in Table 1. We conclude in Section 5.

2 Preliminaries

Table 3. Notation

$[n]$	the set of numbers $\{1, \ldots, n\}$
$s \xleftarrow{\mathsf{R}} \mathcal{S}$	an element s is uniformly at random selected from $\mathcal{S}$
$\mathcal{T} \xleftarrow{\mathsf{R}^i} \mathcal{S}$	i elements are uniformly at random selected from $\mathcal{S}$
$s \xleftarrow{\mathsf{RR}} \mathcal{S}$	an element s is uniformly at random removed from $\mathcal{S}$
$\mathcal{T} \xleftarrow{\mathsf{RR}^i} \mathcal{S}$	i elements are uniformly at random removed from $\mathcal{S}$ and put into $\mathcal{T}$
$\lvert\mathcal{S}\rvert$	the cardinality of the set $\mathcal{S}$
$\mathrm{wt}(\mathbf{v})$, $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_2^n$	$\lvert\{i \mid v_i = 1\}\rvert$, the Hamming weight of $\mathbf{v}$

In the context of this paper, *matrix* always refers to a matrix over $\mathbb{F}_2$. *Addition* and *multiplication* of matrices are ordinary matrix addition and multiplication in which the arithmetic is carried over $\mathbb{F}_2$. Let A and B be two square matrices, we denote by $A \oplus B$ the *direct sum* of A and B which is defined by the block diagonal matrix $\begin{bmatrix} A & \\ & B \end{bmatrix}$.

2.1 Syndrome Decoding Problem

A random binary linear $[n, k, d]$-code $\mathcal{C}$, with block length n, dimension k, and distance d, is a k-dimensional subspace of $\mathbb{F}_2^n$ that can be defined as

$$\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n \mid H\mathbf{c} = \mathbf{0}\},$$

where $H \in \mathbb{F}_2^{(n-k)\times n}$ is random matrix which is known as a *parity check matrix* for the code $\mathcal{C}$. Elements of the code are called *codewords*. The distance of the code d is the minimum weight over $\mathcal{C}$ that is

$$d = \min_{\mathbf{c} \in \mathcal{C}} \{\mathrm{wt}(\mathbf{c}) \mid \mathbf{c} \neq \mathbf{0}\}.$$

The distance d is computed as $d = n\mathsf{H}^{-1}(1 - \frac{k}{n})$, where $\mathsf{H}(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function.

Definition 1 (Syndrome Decoding Problem). *Let $\mathcal{C}$ be a binary linear code whose parity check matrix is $H \in \mathbb{F}_2^{(n-k)\times n}$, and let $\mathbf{s} \in \mathbb{F}_2^{n-k}$ be some given vector.*
The syndrome decoding problem asks for a vector $\mathbf{e} \in \mathbb{F}_2^n$ with weight ω such that

$$H\mathbf{e} = \mathbf{s}.$$

We refer to syndrome decoding instance with parameters n, k and ω as (n, k, ω) syndrome decoding instance.

2.2 Auxiliary Operators

We define two operations of matrices: matrix (or vector) *expansion* and matrix (or vector) *shrinking*.

Expanding Operator Let $\mathbf{x} = (x_1 \dots, x_m) \in \mathbb{F}_2^m$ and let $\mathcal{I}$ be a subset of $[n]$ such that $n > m$ and $|\mathcal{I}| = m$. The vector $\mathbf{x}$ is said to be expanded into the vector $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}_2^n$ using the set $\mathcal{I}$ if each entry x_i of $\mathbf{x}$ is moved to a j^{th} position in $\mathbf{y}$ for $j \in \mathcal{I}$ while maintaining the order of the indices i and j ascendantly. Formally, we define the expansion operation using an operator $\mathsf{E}_{\mathcal{I}}$, and we denote by $\mathbf{y} = \mathsf{E}_{\mathcal{I}}\mathbf{x}$ the operation of expanding $\mathbf{x}$ into $\mathbf{y}$. Thus, the operator $\mathsf{E}_{\mathcal{I}}$ is weight preserving transformation

$$\mathsf{E}_{\mathcal{I}} : \mathbb{F}_2^m \to \mathbb{F}_2^n, n > m$$

which maps its its input to a higher dimensional vector with the same weight.

Shrinking Operators Conversely, for a vector $\mathbf{x} \in \mathbb{F}_2^n$ and a set $\mathcal{I} \subset [n]$, we define row shrinking operator $\mathsf{S}_{\mathsf{r},\mathcal{I}}$ which acts on $\mathbf{x}$ by removing from it the entries indicated by the set $\mathcal{I}$. We write $\mathbf{x} = \mathsf{S}_{\mathsf{r},\mathcal{I}}\mathbf{y}$ to denote shrinking $\mathbf{y}$ into $\mathbf{x}$ using the set $\mathcal{I}$. For a matrix A, the operator $\mathsf{S}_{\mathsf{r},\mathcal{I}}$ shrinks A by removing the rows indicted by the set $\mathcal{I}$ while preserving the order of the remaining rows. Similarly, we define the column shrinking operator $\mathsf{S}_{\mathsf{c},\mathcal{I}}$ which acts on the matrix A by removing the columns indicated by the set $\mathcal{I}$.
For example, when the input is a vector $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{F}_2^n$, the operator

$$\mathsf{S}_{\mathsf{r},\mathcal{I}} : \mathbb{F}_2^n \to \mathbb{F}_2^m, n > m$$

shrinks it by removing the entries $\{a_j \mid j \in \mathcal{I}\}$ while preserving the order of the remaining entries. Likewise, when the input is a matrix $A = [\mathbf{a}_1, \dots, \mathbf{a}_n] \in \mathbb{F}_2^{m \times n}$, the operator $\mathsf{S}_{\mathsf{c},\mathcal{I}}$ removes the columns $\{\mathbf{a}_j \mid j \in \mathcal{I}\}$ while preserving the order of the remaining columns.

It easy to see that

$$\mathsf{S}_{\mathsf{r},\mathcal{I}}A = \mathsf{S}_{\mathsf{c},\mathcal{I}}A^T$$

for every matrix, where A^T is the *transpose* matrix of A.

The shrinking operators when combined with matrix multiplication have some useful properties which we are going to utilize.

The two operators are described in the following two algorithms.

Proposition 1. *Let $\mathbf{x} \in \mathbb{F}_2^m$. Let $\mathcal{I} \subset [n]$ with the complement set $\mathcal{I}^c = [n] \setminus \mathcal{I}$. Then*

$$\mathsf{S}_{\mathsf{r},\mathcal{I}^c}(\mathsf{E}_{\mathcal{I}}\mathbf{x}) = \mathbf{x}. \tag{1}$$

Proof. The proof follows directly from the definitions of the two operators $\mathsf{S}_{\mathsf{r},\mathcal{I}^c}$ and $\mathsf{E}_{\mathcal{I}}$. □

From definition of the operators $\mathsf{S}_{\mathsf{c},\mathcal{I}}$ and $\mathsf{S}_{\mathsf{r},\mathcal{I}}$ and rules of the matrices multiplication the following properties hold.

Algorithm 1 $\mathsf{E}_{\mathcal{I}}$ Expand

```
Require: $\mathbf{a} \in \mathbb{F}_2^m$ and $\mathcal{I} \subset [n]$, $m < n$ such that $|\mathcal{I}| = m$.
Ensure: $\mathbf{y} \in \mathbb{F}_2^n$ such that $\mathrm{wt}(\mathbf{y}) = \mathrm{wt}(\mathbf{a})$.    ▷ Let $\mathbf{a} = (a_1, \cdots, a_m)$ and
    $\mathbf{y} = (y_1, \cdots, y_n)$.
 1: $j \leftarrow 1$
 2:
 3: for $1 \le i \le n$ do
 4:     if $i \in \mathcal{I}$ then
 5:         $y_i \leftarrow a_j$
 6:         $j \leftarrow j + 1$
 7:     else
 8:         $y_i \leftarrow 0$
 9:     end if
10: end for
11: return $\mathbf{y}$
```

Algorithm 2 $\mathsf{S}_{\mathsf{c},\mathcal{I}}$ Shrink

```
Require: $A \in \mathbb{F}_2^{m \times n}$ and a set $\mathcal{I}$ with $|\mathcal{I}| = N$.
Ensure: $A \in \mathbb{F}_2^{m \times (n-N)}$.   ▷ Let $A = \left[\mathbf{a}_1 \cdots \mathbf{a}_n\right]$, where $\mathbf{a}_1 \cdots \mathbf{a}_n$ represent columns of
    $A$.
 1: $j \leftarrow 1$, $i \leftarrow 1$
 2:
 3: while $j \le n$ do
 4:     if $j \notin \mathcal{I}$ then
 5:         $\tilde{\mathbf{a}}_i \leftarrow \mathbf{a}_j$
 6:         $i \leftarrow i + 1$
 7:     end if
 8:     $j \leftarrow j + 1$
 9: end while
10: return $\tilde{A} = \left[\tilde{\mathbf{a}_1} \cdots \tilde{\mathbf{a}}_{n-N}\right]$
```

Matrix Shrinking Properties (MSPs). Let $A = [A_1 \cdots A_n]$ and $B = [B_1 \cdots B_p]$ be two $m \times n$ and $n \times p$, respectively, and let $\mathcal{I}, \mathcal{J} \subset [n]$. Then

(i) $\mathsf{S}_{\mathsf{c},\mathcal{I}}(AB) = A(\mathsf{S}_{\mathsf{c},\mathcal{I}}B)$.

(ii) $\mathsf{S}_{\mathsf{r},\mathcal{J}}(AB) = (\mathsf{S}_{\mathsf{r},\mathcal{J}}A)B$.

(iii) $\mathsf{S}_{\mathsf{c},\mathcal{I}}(\mathsf{S}_{\mathsf{r},\mathcal{J}}(AB)) = \mathsf{S}_{\mathsf{c},\mathcal{I}}((\mathsf{S}_{\mathsf{r},\mathcal{J}}A)B) = (\mathsf{S}_{\mathsf{r},\mathcal{J}}A)(\mathsf{S}_{\mathsf{c},\mathcal{I}}B)$.

(iv) $\mathsf{S}_{\mathsf{r},\mathcal{J}}(\mathsf{S}_{\mathsf{c},\mathcal{I}}(AB)) = \mathsf{S}_{\mathsf{r},\mathcal{J}}(A(\mathsf{S}_{\mathsf{c},\mathcal{I}}B)) = (\mathsf{S}_{\mathsf{r},\mathcal{J}}A)(\mathsf{S}_{\mathsf{c},\mathcal{I}}B)$.

The first two properties are intuitive. The property (iii) follows from applying the operator $\mathsf{S}_{\mathsf{c},\mathcal{I}}$ on both sides of property (ii). Similarly, the property (iv) follows from applying the operator $\mathsf{S}_{\mathsf{r},\mathcal{J}}$ on both sides of property (i).

3 The Basic Scheme: Signing with Errors

Definition 2 (Digital Signature). *A Digital Signature Scheme is a scheme specified by three polynomial-time algorithms* $\mathsf{KeyGen}(1^\lambda)$, $\mathsf{Sign}(pk, m)$ *and,* $\mathsf{Verf}(sk, (m, \boldsymbol{\sigma}))$*, where*

- KeyGen *is a randomized algorithm that on input* 1^λ*, where* λ *is security parameter, returns the pair* (pk, sk) *of public and secret keys.*
- $\mathsf{Sign}(sk, m)$ *is a randomized algorithm that takes the private key* sk *and a message* m *as input and returns the pair* $(m, \boldsymbol{\sigma})$ *of the message* m *and its corresponding signature* $\boldsymbol{\sigma}$*.*
- $\mathsf{Verf}(pk, (m, \boldsymbol{\sigma}'))$ *is a deterministic algorithm that takes as input the public key* pk *and the pair* $(m, \boldsymbol{\sigma}')$ *and response by "Accept" if* $\boldsymbol{\sigma}'$ *is valid signature for the message* m*, or "Reject" otherwise.*

In this section we introduce our basic digital signature algorithm. Then in the next section we develop the basic scheme into an advanced one whose security is based on the hardness of the syndrome decoding problem.

3.1 Generation of the Keys

Algorithm KeyGen 4 is a probabilistic polynomial-time algorithm that on input (n, k, L) returns a pair of (*public*, *private*) keys

$$(pk, sk) = ((H, R, \mathcal{H}, d), (K, \mathbf{r})),$$

where

- $H \in \mathbb{F}_2^{(n-k)\times n}$,
- $R \in \mathbb{F}_2^{(n-k)\times L}$,
- $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^L$ is a secure hash function,
- $d = n\mathsf{H}^{-1}(1 - \frac{k}{n})$, where $\mathsf{H}(x) = -x\log(x) - (1-x)\log(1-x)$,
- $K \in \mathbb{F}_2^{n\times n}$ is nonsingular matrix, and
- $\mathbf{r} \in \mathbb{F}_2^n$ is random vector whose Hamming weight is wt($\mathbf{r}$)=L.

But before presenting the keys generation algorithm KeyGen we first introduce the auxiliary algorithm Init.

Algorithm Init Consider Algorithm Init 3. Steps (9) to (12) in this algorithm generate random n-bits vector $\mathbf{t} = (t_1, \ldots, t_n)$ which has L consecutive 1s within the positions staring from $m_0 + 1$ to $m_0 + m_1$. That is, the vector $\mathbf{t}$ has the following structure

$$t_i = \begin{cases} 0, & 1 \le i \le m_0 \\ 1, & m_0 < i \le m_0 + m_1 \\ 0, & m_0 + m_1 < i \le n \end{cases} \tag{2}$$

In step (12), since the matrix F is random permutation matrix, the vector

$$\mathbf{r} = (r_1, \ldots, r_n) = F\mathbf{t}$$

Algorithm 3 Init

```
Require: $n, d, L$.
Ensure: $(B, M, \mathbf{r})$.
 1: Generate random $n \times n$ permutation matrix $F$
 2: $m_1 \leftarrow L$
 3: $m_2 \leftarrow \left\lceil \frac{n-m_1}{2} \right\rceil$
 4: $m_0 \leftarrow n - (m_1 + m_2)$
 5: Generate random nonsingular matrices $C_i \in \mathbb{F}_2^{m_i \times m_i}$, $i \in \{0, 1, 2\}$
 6: $C \leftarrow C_0 \oplus C_1 \oplus C_2$
 7: $M \leftarrow FCF^{-1}$
 8: $\mathbf{t} = (t_0, \ldots, t_n) \leftarrow (0, \ldots, 0) \in \mathbb{F}_2^n$
 9: for $m_0 + 1 \leq i \leq m_0 + m_1$ do
10:     $t_i \leftarrow 1$
11: end for
12: $\mathbf{r} = (r_1, \ldots, r_n) \leftarrow F\mathbf{t}$
13: Generate random $(n - k) \times n$ matrix $B$
14: return $(B, M, \mathbf{r})$
```

is random vector with randomly distributed weight $\mathrm{wt}(\mathbf{r}) = L$.

Let $\mathcal{T}_{\mathbf{r}}$ denote the set $\{i \mid r_i = 1\}$. Thus, $|\mathcal{T}_{\mathbf{r}}| = L$.

Denote by $\mathcal{T}_{\mathbf{r}}^c$ the set $[n] \setminus \mathcal{T}_{\mathbf{r}}$.

Algorithm KeyGen Algorithm KeyGen 4 generates the pair of the public and private keys (sk, pk).

Algorithm 4 KeyGen The Basic keys Generator

```
Require: $n$, $k$, and $L$.
Ensure: $(pk, sk) = ((H, R, \mathcal{H}, d), (K, \mathbf{r}))$.
 1: $d \leftarrow n\mathsf{H}^{-1}(1 - \frac{k}{n})$        ▷ $\mathsf{H}^{-1}$ is inverse of the binary entropy function.
 2: $(B, M, \mathbf{r}) \leftarrow \mathsf{Init}(n, d, L)$
 3: Generate random $n \times n$ permutation matrix $P$
 4: $K \leftarrow (PM)^{-1}$
 5: $H \leftarrow BK$
 6: $R \leftarrow \mathsf{S}_{\mathsf{c}, \mathcal{T}_{\mathbf{r}}^c} B$
 7: Specify hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \mathbb{F}_2^L$
 8: return $(pk, sk) = ((H, R, \mathcal{H}, d), (K, \mathbf{r}))$
```

3.2 The Basic Signature Algorithm

Let $\mathcal{H}(\cdot)$ be a secure *hash function* such that $\mathcal{H}(\cdot) : \{0,1\}^* \to \{0,1\}^L$. Throughout this work we view the L-bits string $\{0,1\}^L$ as a vector in $\mathbb{F}_2^L$. Precisely, for a message $m \in \{0,1\}^*$, we write $\mathbf{h} = \mathcal{H}(m)$ where $\mathbf{h} \in \mathbb{F}_2^L$ to mean $\mathbf{h}$ is the hash vector (i.e., string) of the message m obtained from the hash function $\mathcal{H}(\cdot)$.

Algorithm 5 $\mathsf{BasicSig}$ The Basic Signature

```
Require: $(m, sk)$, the message $m$ and the private key $sk$.
Ensure: $\mathbf{e}$, the basic signature.
1: $\mathbf{h} \leftarrow \mathcal{H}(m)$
2: $\mathbf{z} = (z_1, \dots, z_n) \leftarrow \mathsf{E}_{\mathcal{T}_\mathbf{r}}\mathbf{h}$
3: $\mathbf{e} \leftarrow K^{-1}\mathbf{z}$
4: return $\mathbf{e}$
```

Consider $\mathsf{BasicSig}$ algorithm 5. Suppose that $\mathbf{h} = (h_1, \dots, h_L)$ and $\mathbf{z} = (z_1, \dots, z_n)$. Observe that $\mathbf{z} = \mathsf{E}_{\mathcal{T}_\mathbf{r}}\mathbf{h}$. Hence, by Proposition 1,

$$\mathbf{h} = \mathsf{S}_{\mathsf{r}, \mathcal{T}_\mathbf{r}^c}\mathbf{z}. \tag{3}$$

Recall that $\mathcal{T}_\mathbf{r} = \{i \mid r_i = 1\}$. Note also from step (2) that $z_i = 0$ for $i \notin \mathcal{T}_\mathbf{r}$. Thus, we may write Step (3) in Algorithm $\mathsf{BasicSig}$ as

$$\begin{aligned}\mathbf{e} = K^{-1}\mathbf{z} &= \sum_{j \in \mathcal{T}_\mathbf{r}} z_j K_j^{-1} \\ &= \sum_{j \in \mathcal{T}_\mathbf{r}} z_j K_j^{-1} \\ &= (\mathsf{S}_{\mathsf{c}, \mathcal{T}_\mathbf{r}^c} K^{-1})(\mathsf{S}_{\mathsf{r}, \mathcal{T}_\mathbf{r}^c}\mathbf{z}) \\ &= U\mathbf{h},\end{aligned}$$

where

$$U = \mathsf{S}_{\mathsf{c}, \mathcal{T}_\mathbf{r}^c} K^{-1} \text{ and } \mathbf{h} = \mathsf{S}_{\mathsf{r}, \mathcal{T}_\mathbf{r}^c}\mathbf{z}. \tag{4}$$

Thus, Algorithm $\mathsf{BasicSig}$ simplifies to

$$\begin{array}{l}\mathsf{BasicSig}(m, sk) \\ \left[\begin{array}{l}\mathbf{h} \leftarrow \mathcal{H}(m) \\ \mathbf{e} \leftarrow U\mathbf{h} \\ \textbf{return } \mathbf{e}\end{array}\right.\end{array} \tag{5}$$

This game notation of the signing operation may be used in the security analysis, since it is concise and neat. For proving correctness of the algorithm, however, it is easier to deal with the original basic algorithm.

Verifying the Basic Signature Simply, a valid signature $\mathbf{e}$ for a message m with hash vector $\mathbf{h}$ must satisfy

$$H\mathbf{e} = \mathbf{s},$$

where $\mathbf{s} = R\mathbf{h}$.

Correctness of the Basic Signature We show that the legitimate signature $\mathbf{e}$ satisfies the identity

$$H\mathbf{e} = \mathbf{s}, \tag{6}$$

where, $H = BK$, $\mathbf{e} = K^{-1}\mathbf{z}$, $\mathbf{z} = \mathsf{E}_{\mathcal{T}_\mathbf{r}}\mathbf{h}$, and $\mathbf{s} = R\mathbf{h}$.

We have $H\mathbf{e} = BKK^{-1}\mathbf{z} = B\mathbf{z}$. We also have $z_j = 0$ for $j \notin \mathcal{T}_\mathbf{r}$, which means $B\mathbf{z} = \sum_{j \in \mathcal{T}_\mathbf{r}} B_j z_j$. That is,

$$\begin{aligned} B\mathbf{z} &= \sum_{j \in \mathcal{T}_\mathbf{r}} B_j z_j \\ &= (\mathsf{S}_{\mathsf{c},\mathcal{T}_\mathbf{r}^c} B)(\mathsf{S}_{\mathsf{r},\mathcal{T}_\mathbf{r}^c}\mathbf{z}) \\ &= R\mathbf{h} = \mathbf{s}, \end{aligned}$$

since $R = \mathsf{S}_{\mathsf{c},\mathcal{T}_\mathbf{r}^c} B$ by definition and $\mathsf{S}_{\mathsf{r},\mathcal{T}_\mathbf{r}^c}\mathbf{z} = \mathbf{h}$ by Equation (3). Hence, the identity $H\mathbf{e} = \mathbf{s}$ holds.

3.3 The Weight of the Basic Signature

Consider Algorithm BasicSig (5). Recall that $M = FCF^{-1}$, where F is $n \times n$ permutation matrix, and $C = \begin{bmatrix} C_0 & & \\ & C_1 & \\ & & C_2 \end{bmatrix}$, where C_i is $m_i \times m_i$ random non-singular matrix, for $i \in \{0, 1, 2\}$. We have

$$\mathbf{e} = K^{-1}\mathbf{z} = PM\mathbf{z} = PFCF^{-1}\mathbf{z}.$$

Let $\mathbf{v} = CF^{-1}\mathbf{z}$. Thus $\mathbf{e} = (PF)\mathbf{v}$. Note that multiplication by the permutation matrix (PF) does not affect the weight. Hence, the wight of the vector $\mathbf{e}$ is equal to the weight of the vector $\mathbf{v}$. In other words, $\mathrm{wt}(\mathbf{e}) = \mathrm{wt}(\mathbf{v})$.
Next, we write $F^{-1}\mathbf{z}$ as

$$F^{-1}\mathbf{z} = (\mathbf{z}_0', \mathbf{z}_1', \mathbf{z}_2') \in \mathbb{F}_2^{m_0} \times \mathbb{F}_2^{m_1} \times \mathbb{F}_2^{m_2}.$$

Recall from Equation (2) that the structure of the vector $\mathbf{t} = F^{-1}\mathbf{r}$, where $\mathbf{t} = (t_1, \ldots, t_n)$ is as follows

$$t_i = \begin{cases} 0, & 1 \le i \le m_0 \\ 1, & m_0 < i \le m_0 + m_1 \\ 0, & m_0 + m_1 < i \le n \end{cases}.$$

It is not hard to observe that the matrix-vector multiplication $F^{-1}\mathbf{r}$ maps (permutes) all of the nonzero entries of the vector $\mathbf{r}$ to the positions starting from $m_0 + 1$ up to $m_0 + m_1$. That is, the matrix F^{-1} acts on $\mathbf{r}$ as a *confining operator* which confines the nonzero entries of $\mathbf{r}$ within the locations from $m_0 + 1$ to $m_0 + m_1$. Since $\{i \mid z_i = 1\} \subseteq \{i \mid r_i = 1\}$, the permutation matrix F^{-1} acts on the vector $\mathbf{z}$ in the same way in which it acts on the vector $\mathbf{r}$. That is, F^{-1} confines the nonzero entries of $\mathbf{z}$ within the positions from $m_0 + 1$ to the position $m_0 + m_1$. Therefore, the vector $F^{-1}\mathbf{z}$ has similar structure as $F^{-1}\mathbf{r}$. In particular, the structure of $F^{-1}\mathbf{z} = (z'_1, \ldots, z'_n)$ will be as follows

$$z'_i = \begin{cases} 0, & \text{for } 1 \le i \le m_0 \\ b \in \{0,1\}, & \text{for } m_0 < i \le m_0 + m_1 \\ 0, & \text{for } m_0 + m_1 < i \le n \end{cases}.$$

Hence, $\mathbf{z}'_0 = \mathbf{0}^{m_0}$ and $\mathbf{z}'_2 = \mathbf{0}^{m_2}$, and

$$\mathbf{v} = \begin{bmatrix} C_0 & & \\ & C_1 & \\ & & C_2 \end{bmatrix} F^{-1}\mathbf{z} = \begin{bmatrix} C_0 & & \\ & C_1 & \\ & & C_2 \end{bmatrix} \begin{bmatrix} \mathbf{z}'_0 \\ \mathbf{z}'_1 \\ \mathbf{z}'_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0}^{m_0} \\ C_1\mathbf{z}'_1 \\ \mathbf{0}^{m_2} \end{bmatrix}. \tag{7}$$

Since C_1 is $m_1 \times m_1$ random matrix, we estimate the weight of the vector $\mathbf{v}$ as

$$\mathrm{wt}(\mathbf{v}) \approx \frac{m_1}{2}.$$

Remark 1. Observe that C_1 is full-rank random matrix and $\mathbf{z}'_1$ is random vector. Therefore, entries of the vector $C_1\mathbf{z}'_1$ are random where each entry is 1 or 0 with probability $\frac{1}{2}$.

Since $\mathbf{e} = (PF)\mathbf{v}$ and each of F and P is a permutation matrix and hence it preserves the weight, the weight of the vector $\mathbf{e}$ is equal to weight of $\mathbf{v}$. That is,

$$\mathrm{wt}(\mathbf{e}) \approx \frac{m_1}{2} \approx \frac{L}{2},$$

since $m_1 = L$. However, the multiplication of $\mathbf{v}$ by F and then by P redistributes its weight uniformly, since both of F and P are uniform random. Therefore, unlike the weight of the vector $\mathbf{v}$, the weight of the vector $\mathbf{e}$ is uniformly distributed over its n coordinates.

3.4 Structure of the Basic Signature

It is not hard to observe that the basic signature $\mathbf{e}$ has a specific structure. We show this as follows. Let

$$\mathbf{e}^\star = (e^\star_1, \ldots, e^\star_n) = PF\mathbf{t},$$

where $\mathbf{t} = (t_1, \ldots, t_n)$ with $t_i = 0$ for $i \notin \{m_0+1, \ldots, m_0+m_1\}$ and $t_i = 1$ for $i \in \{m_0+1, \ldots, m_0+m_1\}$. Then consider structure of the vector $\mathbf{v} = CF^{-1}\mathbf{z}$ as shown in Equation (7) where $\mathbf{v} = (v_1, \ldots, v_n)$ with $v_i = 0$ for $i \notin \{m_0 + 1, \ldots, m_0+m_1\}$ and (unlike $\mathbf{t}$) $v_i \in \{0,1\}$ for $i \in \{m_0+1, \ldots, m_0+m_1\}$. Since

$$\mathbf{e} = (e_1, \ldots, e_n) = PF\mathbf{v},$$

it follows that, for every $\mathbf{e}$,

$$\{i \mid e_i = 1\} \subseteq \{i \mid e^\star_i = 1\}.$$

Next, let $\mathcal{T}_{\mathbf{e}^\star} = \{i \mid e_i^\star = 1\}$. Observe that $|\mathcal{T}_{\mathbf{e}^\star}| = m_1 = L$. Thus, for every signature $\mathbf{e} = (e_1, \ldots, e_n)$,

$$e_i = 0 \text{ for } i \notin \mathcal{T}_{\mathbf{e}^\star}.$$

The possible nonzero entries of $\mathbf{e}$ have the same probability. That is,

$$Pr[e_j = 1 \mid j \in \mathcal{T}_{\mathbf{e}^\star}] = \frac{1}{2}, \tag{8}$$

because the sub-matrix C_1 is full-rank random matrix and the hash vector $\mathbf{h}$ is random (see Remark 1).

So, when checking if $H\mathbf{e} = \mathbf{s}$ holds, the columns of the matrix H that correspond to the zero entries in $\mathbf{e}$ (i.e., the columns $\{H_j \mid j \notin \mathcal{T}_{\mathbf{e}^\star}\}$) are not involved in the verification. Consequently, we may rewrite the verification equation of the basic signature (6) as follows

$$H\mathbf{e} = \sum_{j \in \mathcal{T}_{\mathbf{e}^\star}} H_j e_j = \mathbf{s}. \tag{9}$$

In the advanced scheme the basic signature $\mathbf{e}$ is hidden as well as the set $\mathcal{T}_{\mathbf{e}^\star}$ by adding $\mathbf{e}$ to a secret vector $\mathbf{c}$ and the resulting sum gives us the advanced signature. Luckily, the verification equation remain as before except for a slight modification by adding a restriction on the weight of the signature. Precisely, the new signature is $(\boldsymbol{\sigma}, \varsigma) \in \mathbb{F}_2^n \times \mathbb{F}_2^\theta$ with $\mathrm{wt}(\boldsymbol{\sigma}) = d$ such that

$$Pr[\sigma_i = 1] > 0 \text{ for all } i \in [n].$$

The set $\mathcal{T}_{\mathbf{e}^\star}$ which has size L will be included within a larger set such that it is statistically hard to distinguish it.

4 An Advanced Scheme

In this section we develop the basic signature scheme further into an advanced one by randomizing the signature. To this end, we modify the keys of the basic scheme and define an advanced signing algorithm that uses the basic signing algorithm as a subroutine. More precisely, we modify the keys by modifying some columns of the matrix H to make it *act like* a parity check matrix for a random binary linear code. Then we modify the private key accordingly. The verification remain almost unchanged but we put a restriction on the weight of the new signature.

Modifying the Keys Consider the public key of the basic scheme $pk = (H, \mathcal{H}, R, d)$. Recall that $H \in \mathbb{F}_2^{(n-k)\times n}$ and $R \in \mathbb{F}_2^{(n-k)\times L}$. Let $T \in \mathbb{F}_2^{\theta\times l}$ be uniformly random matrix. We simultaneously modify H into a new matrix $H^\star$ and generate a random matrix $G \in \mathbb{F}_2^{n\times l}$ such that for every column G_c of the matrix G, $H^\star G_c = \mathbf{b}_c$, where $\mathbf{b}_c = \mathbf{a}_i + \sum_{k=1}^{\theta} R_k T_{k,c}$ for a random secret vector

$\mathbf{a}_i \in \{\mathbf{a}_1, \ldots, \mathbf{a}_\theta\}$ with $\sum_k^\theta \mathbf{a}_i = \mathbf{0}$. However, the columns $\{H_j \mid j \in \mathcal{T}_{\mathbf{e}^\star}\}$ remain unchanged. The modifications occur only among the columns $\{H_j \mid j \notin \mathcal{T}_{\mathbf{e}^\star}\}$. We set the public key for the advanced scheme as $pk^\star = (H^\star, \mathcal{H}, R, d, \theta)$. That is, H is replaced with $H^\star$ in the new public key. We update the private key sk to $sk^\star$ accordingly by combining the matrices G and T with the basic private key sk.

The keys of the advanced scheme are generated by Algorithm $\mathsf{KeyGen}^\star$ 6.

The Advanced Signature To sign a message m using the advanced signature scheme, we first obtain the basic signature $\mathbf{e}$ of m using the basic signing algorithm $\mathsf{BasicSig}$ 5, then a random set $\mathcal{J}$ generated according to the set $\{i \in \mathcal{T}_{\mathbf{e}^\star} \mid e_i = 1\}$. The new signature $(\boldsymbol{\sigma}, \boldsymbol{\varsigma}) \in \mathbb{F}_2^n \times \mathbb{F}_2^\theta$ is obtained as

$$\boldsymbol{\sigma} = \sum_{c \in \mathcal{J}} G_c + \mathbf{e} \text{ and } \boldsymbol{\varsigma} = \sum_{c \in \mathcal{J}} R_k T_{k,c}$$

such that the Hamming weight of the vector $\boldsymbol{\sigma}$ is exactly d. The detailed description of the signing operation is given in Algorithm Sign 7.

The Verification of the Signature The verification of a valid signature $\boldsymbol{\sigma}$ for a message m is performed as follows.
Accept the signature $(\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n), \boldsymbol{\varsigma} = (\varsigma_1, \ldots, \varsigma_\theta))$ if and only if

$$H^\star \boldsymbol{\sigma} = \mathbf{s}^\star \text{ and } \mathrm{wt}(\boldsymbol{\sigma}) = d,$$

where $\mathbf{s}^\star = \mathbf{s} + \sum_{k=1}^\theta R_k \varsigma_k$, $\mathbf{s} = R\mathbf{h}$, and $\mathbf{h} = \mathcal{H}(m)$ is the hash vector (i.e., string) of the message m.

Thus, for a particular message m with hash vector $\mathbf{h}$, forging a signature $(\boldsymbol{\sigma}, \boldsymbol{\varsigma}) \in \mathbb{F}_2^n \times \mathbb{F}_2^\theta$ is equivalent to finding a vector $\boldsymbol{\sigma}$ that solves the syndrome decoding instance (n, k, d) using $H^\star$ as a parity-check matrix.

Now it is time to delve into the algorithms $\mathsf{KeyGen}^\star$ and Sign in full details.

Algorithm $\mathsf{KeyGen}^\star$ The algorithm $\mathsf{KeyGen}^\star(pk, sk, \mathcal{T}_{\mathbf{e}^\star}, d, L, \epsilon)$ takes as input the public and the private keys of the basic signature scheme (pk, sk) along with the set $\mathcal{T}_{\mathbf{e}^\star}$ and the parameters d, L and ϵ and generates the pair $(pk^\star, sk^\star)$ of the public and private keys of the advanced signature scheme, where

$$pk^\star = (H^\star, \mathcal{H}, R, d, \theta) \text{ and } sk^\star = (sk, S_1, \ldots, S_\theta, G, T),$$

where $H^\star \in \mathbb{F}_2^{(n-k)\times k}$ with $H_j^\star = H_j$ for $j \in \mathcal{T}_{\mathbf{e}^\star}$. And, S_i is sorted random subset of $\mathcal{T}_{\mathbf{e}^\star}$ with $|S_i| = \epsilon$, for $1 \le i \le \theta$, $S_i \cap S_{i'} = \emptyset$ for $i \neq i'$, and $\underset{1 \le i \le \theta}{\cup} S_i = \mathcal{T}_{\mathbf{e}^\star}$. The matrix $G \in \mathbb{F}_2^{n\times l}$ is generated in accordance with the matrix $H^\star$, and $T \in \mathbb{F}_2^{\theta\times l}$

is random independent matrix, where $l = \theta 2^\epsilon$ and $\theta = \frac{L}{\epsilon}$.

Algorithm 6 $\mathsf{KeyGen}^\star$ Advanced Keys Generation

```
Require: $pk, sk, \mathcal{T}_{\mathbf{e}^\star}, d, L, \epsilon$.
Ensure: $(pk^\star, sk^\star)$.
 1: $H^\star \leftarrow H$
 2: $\theta \leftarrow \frac{L}{\epsilon}$, $l \leftarrow \theta 2^\epsilon$
 3: $\mathcal{N} \xleftarrow{\mathsf{RR}^{\theta 2^\epsilon}} [n] \setminus \mathcal{T}_{\mathbf{e}^\star}$
 4: $\mathcal{N}^\star \leftarrow [n] \setminus \mathcal{N}$
 5: $T \xleftarrow{\mathsf{R}} \mathbb{F}_2^{\theta \times l}$
 6:
 7: for $1 \leq k \leq \theta$ do
 8:     $\mathcal{N}_k \xleftarrow{\mathsf{RR}^{2^\epsilon}} \mathcal{N}$
 9:     $\mathcal{N}_k^\star \xleftarrow{\mathsf{RR}^{2^\epsilon}} \mathcal{N}^\star$
10: end for
11: $\{\mathbf{a}_1, \ldots, \mathbf{a}_{\theta-1}\} \xleftarrow{\mathsf{R}^{\theta-1}} \mathbb{F}_2^{(n-k)}$
12: $\mathbf{a}_\theta \leftarrow \sum_{k=1}^{(\theta-1)} \mathbf{a}_k$
13: Choose random numbers $\tau_1, \ldots, \tau_\theta$ such that $\sum_{i=1}^{\theta} \tau_i = d$    ▷ We choose $\tau_i \approx \frac{d}{\theta}$.
14:
15: for $1 \leq i \leq \theta$ do
16:     $\mathcal{T}_i = \{s_{i1}, \ldots, s_{i\epsilon}\} \xleftarrow{\mathsf{RR}^\epsilon} \mathcal{T}_{\mathbf{e}^\star}$
17:     $S_i \leftarrow (s_{i1}, \ldots, s_{i\epsilon})$    ▷ with $s_{ik} < s_{i(k+1)}$, $1 \leq k < \epsilon$.
18:
19:     for $0 \leq j \leq 2^\epsilon - 1$ do    ▷ where $j = (j_1, \ldots, j_\epsilon)_2$
20:         $c \leftarrow (i-1)2^\epsilon + j$, $\alpha_c \xleftarrow{\mathsf{RR}} \mathcal{N}_i$
21:         $\mathbf{b}_c \leftarrow \mathbf{a}_i + \sum_{k=1}^{\theta} R_k T_{k,c}$
22:         $\mathcal{O}_c \leftarrow \{s_{ik} \mid j_k = 1\}$
23:
24:         if $j > 0$ then
25:             $\mathcal{P}_c \xleftarrow{\mathsf{R}^{\tau_i - 2}} \mathcal{N}_i^\star \setminus \mathcal{O}_c$
26:             $\alpha_c' \xleftarrow{\mathsf{R}} \{\alpha_{c-1}, \ldots, \alpha_{c-j}\}$
27:             $\mathcal{Q}_c \leftarrow \{\alpha_c, \alpha_c'\}$
28:         else
29:             $\mathcal{P}_c \xleftarrow{\mathsf{R}^{\tau_i - 1}} \mathcal{N}_i^\star \setminus \mathcal{O}_c$
30:             $\mathcal{Q}_c \leftarrow \{\alpha_c\}$
31:         end if
32:         $\mathcal{I}_c \leftarrow \{\mathcal{O}_c \cup \mathcal{P}_c \cup \mathcal{Q}_c\}$
33:         $H_{\alpha_c}^\star \leftarrow \mathbf{b}_c + \sum_{k \in \{\mathcal{I}_c \setminus \alpha_c\}} H_k^\star$
34:
35:         for $r \in \mathcal{I}_c$ do
36:             $G_{r,c} \leftarrow 1$
37:         end for
38:     end for
39: end for
40: $pk^\star \leftarrow (H^\star, \mathcal{H}, R, d, \theta)$
41: $sk^\star \leftarrow (sk, S_1, \ldots, S_\theta, G, T)$
42: return $(pk^\star, sk^\star)$
```

Now, observe that in step (33) of Algorithm $\mathsf{KeyGen}^\star$ 6 the only columns that get changed are columns $H_{\alpha_c}^\star$ where $\alpha_c \notin \mathcal{T}_{\mathbf{e}^\star}$, since $\alpha_c \in \mathcal{N}$ and $\mathcal{N} \cap \mathcal{T}_{\mathbf{e}^\star} = \emptyset$. Thus,

$$H_j^\star = H_j \text{ for } j \in \mathcal{T}_{\mathbf{e}^\star}, \tag{10}$$

since the columns $\{H_j^\star \mid j \in \mathcal{T}_{\mathbf{e}^\star}\}$ do not change. Observe also from the step (33) that

$$\sum_{k \in \mathcal{I}_c} H_k^\star = \mathbf{b}_c.$$

Next, note from the loop in steps (35) to (37) that

$$G_{r,c} = \begin{cases} 1, & \text{for } r \in \mathcal{I}_c \\ 0, & \text{otherwise} \end{cases},$$

for every c^{th} column of G, $1 \leq c \leq l$. Hence

$$H^{\star}G_c = \mathbf{b}_c, \tag{11}$$

where G_c denotes the c^{th} column of the matrix G, $c = (i-1)2^{\epsilon} + j$. And, consequently,

$$H^{\star}G_c = \mathbf{a}_i + \sum_{k=1}^{\theta} R_k T_{k,c}, \tag{12}$$

for every $c = (i-1)2^{\epsilon} + j$, as $\mathbf{b}_c = \mathbf{a}_i + \sum_{k=1}^{\theta} R_k T_{k,c}$, where R_k is the k^{th} column of the matrix R.

4.1 Algorithm Sign

To sign the message m using the private key $sk^{\star}$ the Algorithm Sign works as follows. It obtains the basic signature $\mathbf{e} = \mathsf{BasicSig}(m, sk)$. Then an empty set $\mathcal{J}$ is created, and for every $S_i = (s_{i1}, \ldots, s_{i\epsilon})$, $1 \leq i \leq \theta$, the algorithm computes $c = (i-1)2^{\epsilon} + j$ where $j = \sum_{k=1}^{\epsilon} e_{s_{ik}} 2^{k-1}$; that is j has the binary representation $j = (e_{s_{i1}}, ..., e_{s_{i\epsilon}})_2$. Every time the number c is added to the set $\mathcal{J}$. Next, the algorithm computes $\mathbf{c} = \sum_{c \in \mathcal{J}} G_c$ and finally the advanced signature $(\boldsymbol{\sigma}, \boldsymbol{\varsigma})$ is obtained as $\boldsymbol{\sigma} = \mathbf{c} + \mathbf{e}$ and $\boldsymbol{\varsigma} = \sum_{c \in \mathcal{J}} T_c$.

Algorithm 7 Sign The Advanced Signing Algorithm

```
Require: $(m, sk^{\star})$–the message $m$ and the private key $sk$.
Ensure: $(m, \boldsymbol{\sigma}, \boldsymbol{\varsigma})$– the message $m$ and the signature $(\boldsymbol{\sigma}, \boldsymbol{\varsigma})$.
 1: $\mathbf{e} \leftarrow \mathsf{BasicSig}(m, sk)$
 2: $\mathcal{J} \leftarrow \emptyset$
 3:
 4: for $1 \leq i \leq \theta$ do
 5:     Fetch $S_i = (s_{i1}, \ldots, s_{i\epsilon})$
 6:     $j \leftarrow \sum_{k=1}^{\epsilon} e_{s_{ik}} 2^{k-1}$    ▷ $j = (e_{s_{i1}}, ..., e_{s_{i\epsilon}})_2$
 7:     $c \leftarrow (i-1)2^{\epsilon} + j$
 8:     $\mathcal{J} \leftarrow \mathcal{J} \cup c$
 9: end for
10: $\mathbf{c} \leftarrow \sum_{c \in \mathcal{J}} G_c$
11: $\boldsymbol{\sigma} \leftarrow \mathbf{c} + \mathbf{e}$
12: $\boldsymbol{\varsigma} \leftarrow \sum_{c \in \mathcal{J}} T_c$
13: return $(m, \boldsymbol{\sigma}, \boldsymbol{\varsigma})$
```

From Algorithm Sign 7 and Equation (12) it is easy to see that

$$\begin{aligned} H^{\star}\mathbf{c} &= H^{\star} \sum_{c \in \mathcal{J}} G_c = \sum_{c \in \mathcal{J}} H^{\star} G_c \\ &= \sum_{c \in \mathcal{J}} \mathbf{b}_c \\ &= \sum_{c \in \mathcal{J}} \mathbf{a}_i + \sum_{c \in \mathcal{J}} \sum_{k=1}^{\theta} R_k T_{k,c}. \end{aligned}$$

Note that $\sum_{k=1}^{\theta} \mathbf{a}_i = \mathbf{0}$ from Algorithm 6 step (11). Note also that

$$\sum_{c\in\mathcal{J}} \sum_{k=1}^{\theta} R_k T_{k,c} = \sum_{k=1}^{\theta} R_k \sum_{c\in\mathcal{J}} T_{k,c}.$$

Thus,

$$H^{\star}\mathbf{c} = \sum_{k=1}^{\theta} R_k \sum_{c\in\mathcal{J}} T_{k,c}. \tag{13}$$

Also, from Algorithm Sign 7 step (12) we have $\boldsymbol{\varsigma} = (\varsigma_1, \ldots, \varsigma_\theta) = \sum_{c\in\mathcal{J}} T_c$. Thus

$$\varsigma_k = \sum_{c\in\mathcal{J}} T_{k,c}, \text{ for } 1 \le k \le \theta. \tag{14}$$

(From the context, it is obvious for the reader that $\boldsymbol{\varsigma} = (\varsigma_1, \ldots, \varsigma_\theta)$ is a column (not row) vector although it written as a row vector.)
Substituting $\sum_{c\in\mathcal{J}} T_{k,c} = \varsigma_k$ in Equation (13) we get

$$H^{\star}\mathbf{c} = \sum_{k=1}^{\theta} R_k \varsigma_k. \tag{15}$$

Proposition 2. *Let* $\mathbf{e} = (e_1, \ldots, e_n)$ *be a particular basic signature. Consider the set* $\mathcal{J}$ *which is generated by Algorithm* Sign *7 according to* $\mathbf{e}$*. Then*

$$\{s \mid e_s = 1\} = \underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c.$$

Proof. It is obvious that

$$\{s \mid e_s = 1\} = \{s \in \mathcal{T}_{\mathbf{e}^{\star}} \mid e_s = 1\}$$

since $e_s = 0$ for $s \notin \mathcal{T}_{\mathbf{e}^{\star}}$.

Next, consider Algorithm Sign 7. Note that $|\mathcal{J}| = \theta$. For every $c \in \mathcal{J}$ where $c = (i-1)2^{\epsilon} + j$, we have $j = (e_{s_{i1}}, \ldots, e_{s_{i\epsilon}})_2$ where $\le i \le \theta$. Now, we going to show that for every $c \in \mathcal{J}$,

$$\mathcal{O}_c = \{s_{ik} \mid e_{s_{ik}} = 1, 1 \le k \le \epsilon\}$$

and therefore, by taking the union $\underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c$ the proposition follows.

To see this observe that (from steps (17), (20), and (22) of Algorithm 6) for every tuple $S_i = (s_{i1}, \ldots, s_{i\epsilon})$ and every $c = (i-1)2^{\epsilon} + j$, where j has the binary representation $j = (j_1, \ldots, j_\epsilon)_2$, $\mathcal{O}_c = \{s_{ik} \mid j_k = 1\}$. Since $\mathcal{O}_c = \{s_{ik} \mid j_k = 1\}$ and in step (6) of Sign 7, $j = (j_1, \ldots, j_\epsilon)_2 = (e_{s_{i1}}, \ldots, e_{s_{i\epsilon}})_2$, it follows that $\mathcal{O}_c = \{s_{ik} \mid e_{s_{ik}} = 1\}$, for every $c \in \mathcal{J}$. Thus

$$\underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c = \underset{1\le i\le\theta}{\cup} \{s_{ik} \mid e_{s_{ik}} = 1, 1 \le k \le \epsilon\}.$$

Note that (from step (16) of Algorithm $\mathsf{KeyGen}^\star$ 6) $\underset{1\le i\le\theta}{\cup}\{s_{ik}, 1 \le k \le \epsilon\} = \mathcal{T}_{\mathbf{e}^\star}$. Therefore,

$$\underset{c\in\mathcal{J}}{\cup}\mathcal{O}_c = \{s \in \mathcal{T}_{\mathbf{e}^\star} \mid e_s = 1\} = \{s \mid e_s = 1\}. \tag{16}$$

and the proof completes. □

The Weight of the Vector $\boldsymbol{\sigma}$ Here we show that $\mathrm{wt}(\boldsymbol{\sigma}) = d$.

Consider the matrix $G = [G_1 \cdots G_l]$, where G_c, $1 \le c \le l$ is the c^{th} column of G. More precisely, the matrix G is defined by Algorithm $\mathsf{KeyGen}^\star$ 6 as follows

$$G_{r,c} = \begin{cases} 1, & \text{for } r \in \mathcal{I}_c \\ 0, & \text{for } r \notin \mathcal{I}_c \end{cases},$$

where $\mathcal{I}_c = \{\mathcal{O}_c \cup \mathcal{P}_c \cup \mathcal{Q}_c\}$ is a random set drawn from $[n]$ in steps (22) to (30).

Suppose that a vector $\boldsymbol{\sigma}$ is generated by Algorithm Sign 7 as

$$\boldsymbol{\sigma} = \mathbf{c} + \mathbf{e} = \sum_{c\in\mathcal{J}} G_c + \mathbf{e},$$

where $\mathbf{c} = (c_1, \ldots, c_n)$ and $\mathbf{e} = (e_1, \ldots, e_n)$.
The weight of the vector $\boldsymbol{\sigma} = \sum_{c\in\mathcal{J}} G_c + \mathbf{e}$ is determined by the number and the distribution of the nonzero entries $\{G_{r,c}, c \in \mathcal{J} \mid G_{r,c} = 1\}$ and $\{e_r \mid e_r = 1\}$. Now, the number of the nonzero entries in the positions $G_{r,c}$, $c \in \mathcal{J}$ is

$$|\{G_{r,c}, c \in \mathcal{J} \mid G_{r,c} = 1\}| = \sum_{c\in\mathcal{J}} (|\mathcal{I}_c| = |\{\mathcal{O}_c \cup \mathcal{P}_c \cup \mathcal{Q}_c\}|).$$

Note that $\mathcal{O}_c \cap \mathcal{P}_c = \emptyset$ and $\{\mathcal{O}_c \cup \mathcal{P}_c\} \cap \mathcal{Q}_c = \emptyset$ for every c. Note also that $\mathcal{O}_c \cap \mathcal{O}_{c'} = \emptyset$ for every $c \neq c'$. Moreover, it is not hard to see that $\{\mathcal{P}_c \cup \mathcal{Q}_c\} \cap \{\mathcal{P}_{c'} \cup \mathcal{Q}_{c'}\} = \emptyset$ for every $\{c, c'\} \subset \mathcal{J}$, $c \neq c'$. Therefore,

$$\begin{aligned} |\{G_{r,c}, c \in \mathcal{J} \mid G_{r,c} = 1\}| \quad &= |\{G_{r,c} = 1 \mid r \in \underset{c\in\mathcal{J}}{\cup}\mathcal{O}_c\}| \\ &+ |\{G_{r,c} = 1 \mid r \in \underset{c\in\mathcal{J}}{\cup}\{\mathcal{P}_c \cup \mathcal{Q}_c\}| \end{aligned} \tag{17}$$

which means

$$|\{G_{r,c}, c \in \mathcal{J} \mid G_{r,c} = 1\}| = \sum_{c\in\mathcal{J}} |\mathcal{O}_c| + \sum_{c\in\mathcal{J}} |\{\mathcal{P}_c \cup \mathcal{Q}_c\}|. \tag{18}$$

Let $\delta = \sum_{c\in\mathcal{J}} |\mathcal{O}_c|$. We have, from Algorithm 6 steps (24) to (31), $|\{\mathcal{P}_c \cup \mathcal{Q}_c\}| = \tau_i$ for every c. And, also from Algorithm 6 step (13) we see that $\sum_{i=1}^{\theta} \tau_i = d$. Hence, by substituting in Equation (18) we find

$$|\{G_{r,c}, c \in \mathcal{J} \mid G_{r,c} = 1\}| = \delta + d.$$

Next, we have $\underset{c\in\mathcal{J}}{\cup}\mathcal{O}_c = \{s \mid e_s = 1\}$ by Proposition 2. Hence, $G_{r,c} = 1$ for $r \in \underset{c\in\mathcal{J}}{\cup}\mathcal{O}_c$ implies

$$G_{r,c} = 1 \text{ for } r \in \{s \mid e_s = 1\}, c \in \mathcal{J}.$$

Therefore, for every r where $e_r = 1$ there exists $c \in \mathcal{J}$ such that $G_{r,c} = 1$. Further,

$$|\underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c| = \sum_{c\in\mathcal{J}} |\mathcal{O}_c| = \delta, \text{ and hence } |\{s \mid e_s = 1\}| = \delta.$$

That is, $\mathrm{wt}(\mathbf{e}) = \delta$.
Now, when performing the sum

$$\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n) = \mathbf{c} + \mathbf{e} = \sum_{c\in\mathcal{J}} G_c + \mathbf{e}$$

every nonzero entry of the vector $\mathbf{e}$ (i.e., every entry in $\{e_r \mid e_r = 1\}$) sums to zero with the corresponding nonzero entry in the set $\{G_{r,c} = 1 \mid r \in \underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c\}$. As result, exactly δ nonzero entries from the matrix G vanish by adding the vector $\mathbf{c} = \sum_{c\in\mathcal{J}} G_c$ to the vector $\mathbf{e}$, and the corresponding δ nonzero entries of $\mathbf{e}$ vanish as well, because

$$|\{G_{r,c} = 1 \mid r \in \underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c\}| = |\{e_r \mid e_r = 1\}| = \delta.$$

Therefore, when finding the weight of the vector $\boldsymbol{\sigma} = \sum_{c\in\mathcal{J}} G_c + \mathbf{e}$ in terms of the nonzero entries $\{G_{r,c}, c \in \mathcal{J} \mid G_{r,c} = 1\}$ and $\{e_r \mid e_r = 1\}$ the term $|\{G_{r,c} = 1 \mid r \in \underset{c\in\mathcal{J}}{\cup} \mathcal{O}_c\}|$ of Equation (17) which equals δ cancels out with $|\{e_r \mid e_r = 1\}|$.
We have $\{\mathcal{P}_c \cup \mathcal{Q}_c\} \cap \{\mathcal{P}_{c'} \cup \mathcal{Q}_{c'}\} = \emptyset$ for every $\{c, c'\} \subset \mathcal{J}$, $c \neq c'$. Therefore,

$$\mathrm{wt}(\boldsymbol{\sigma}) = |\{r \mid \sigma_r = 1\}| = |\{G_{r,c} = 1 \mid r \in \underset{c\in\mathcal{J}}{\cup} \{\mathcal{P}_c \cup \mathcal{Q}_c\}|.$$

That is,

$$\mathrm{wt}(\boldsymbol{\sigma}) = \sum_{c\in\mathcal{J}} |\{\mathcal{P}_c \cup \mathcal{Q}_c\}| = d. \tag{19}$$

Algorithm 8 Verf Verifying Algorithm

```
Require: $(m, \boldsymbol{\sigma}, \varsigma, pk^\star)$.
Ensure: Accept or reject the the signature $(\boldsymbol{\sigma}, \varsigma)$.
1: $\mathbf{h} \leftarrow \mathcal{H}(m)$, $\mathbf{s} \leftarrow R\mathbf{h}$
2: $\mathbf{s}^\star \leftarrow \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k$
3: if $\mathrm{wt}(\boldsymbol{\sigma}) = d$ and $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$ then
4:     Accept
5: else
6:     Reject
7: end if
```

4.2 Correctness of the Advanced Scheme

Suppose that we have the pair of the message and its signature $(m, (\boldsymbol{\sigma}, \varsigma))$. We have already seen from Equation (19) that for every message m the algorithm Sign will generate a vector $\boldsymbol{\sigma}$ that satisfies $\mathrm{wt}(\boldsymbol{\sigma}) = d$. In what follows we show that $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$, where $\mathbf{s}^\star = \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k$.
We know that the basic signature satisfies $H\mathbf{e} = \mathbf{s}$ and we also know that $e_j = 0$ for $j \notin \mathcal{T}_{\mathbf{e}^\star}$. Hence, $H\mathbf{e} = \sum_{j \in \mathcal{T}_{\mathbf{e}^\star}} H_j e_j = \mathbf{s}$. Consequently,

$$H^\star \mathbf{e} = \sum_{j \in \mathcal{T}_{\mathbf{e}^\star}} H_j^\star e_j = \sum_{j \in \mathcal{T}_{\mathbf{e}^\star}} H_j e_j = \mathbf{s},$$

where $H_j^\star = H_j$ for $i \in \mathcal{T}_{\mathbf{e}^\star}$ from Equation (10).
Next, let us recall Equation (15)

$$H^\star \mathbf{c} = \sum_{k=1}^{\theta} R_k \varsigma_k.$$

Now, since $\boldsymbol{\sigma} = \mathbf{c} + \mathbf{e} = \mathbf{e} + \mathbf{c}$,

$$\begin{aligned} H^\star \boldsymbol{\sigma} &= H^\star \mathbf{e} + H^\star \mathbf{c} \\ &= \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k \\ &= \mathbf{s}^\star. \end{aligned}$$

Thus, the identity $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$ holds.

4.3 Indistinguishability of the Set $\mathcal{T}_{\mathbf{e}^\star}$

In the basic signature $\mathbf{e}$ the random set $\mathcal{T}_{\mathbf{e}^\star}$ is noticeable since for every basic signature $\mathbf{e} = (e_1, \ldots, e_n)$,

$$Pr[e_i = 1 \mid i \in \mathcal{T}_{\mathbf{e}^\star}] = \frac{1}{2} \text{ whereas } Pr[e_i = 1 \mid i \notin \mathcal{T}_{\mathbf{e}^\star}] = 0.$$

As we see, this problem has been resolved in the advanced scheme by adding a vector $\mathbf{c}$ to the basic signature and hence the new signature is randomized such that $Pr[\sigma_i = 1] > 0$ for all $i \in [n]$. Moreover, it is statistically hard to distinguish the set of positions $\mathcal{T}_{\mathbf{e}^\star}$.
To see this, consider the two sets $\mathcal{N}$ and $\mathcal{N}^\star$ from Algorithm 6. It is easy to see from the steps (3) and (4) that

$$\mathcal{N} \cap \mathcal{N}^\star = \emptyset \text{ and } \mathcal{N} \cup \mathcal{N}^\star = [n].$$

Note that $\mathcal{T}_{\mathbf{e}^\star} \subset \mathcal{N}^\star$.
Now, for every vector $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ with $\mathrm{wt}(\boldsymbol{\sigma}) = d$, we have from Equation (19)

$$\mathrm{wt}(\boldsymbol{\sigma}) = \sum_{c \in \mathcal{J}} |\{\mathcal{P}_c \cup \mathcal{Q}_c\}| = \sum_{c \in \mathcal{J}} |\mathcal{P}_c| + \sum_{c \in \mathcal{J}} |\mathcal{Q}_c| = d$$

Since $|\mathcal{P}_c| = \tau_i - a$, $a \in \{1, 2\}$ for every c,

$$|\{r \in \underset{c\in\mathcal{J}}{\cup} \mathcal{P}_c \mid \sigma_r = 1\}| = \sum_{c\in\mathcal{J}} |\mathcal{P}_c| = d - \vartheta \text{ such that } \theta \le \vartheta \le 2\theta.$$

We have $\underset{c\in\mathcal{J}}{\cup} \mathcal{P}_c \subset \mathcal{N}^\star$. Hence, $|\{r \in \mathcal{N}^\star \mid \sigma_r = 1\}| = d - \vartheta$. And,

$$|\{r \in \underset{c\in\mathcal{J}}{\cup} \mathcal{Q}_c \mid \sigma_r = 1\}| = \sum_{c\in\mathcal{J}} |\mathcal{Q}_c| = \vartheta$$

where $\underset{c\in\mathcal{J}}{\cup} \mathcal{Q}_c \subset \mathcal{N}$ and hence $|\{r \in \mathcal{N}^\star \mid \sigma_r = 1\}| = \vartheta$.
Note that $|\mathcal{N}| = |\mathcal{N}^\star| = \theta 2^\epsilon$. Therefore,

$$Pr[\sigma_r = 1] = \begin{cases} p^\star, \; r \in \mathcal{N}^\star \\ p, \;\; r \in \mathcal{N} \end{cases}, \tag{20}$$

where $\frac{d-2\theta}{\theta 2^\epsilon} \le p^\star \le \frac{d-\theta}{\theta 2^\epsilon}$ and $\frac{\theta}{\theta 2^\epsilon} \le p \le \frac{2\theta}{\theta 2^\epsilon}$ For the selected security parameters d, ϵ and θ the concrete values of these probabilities are within the ranges

$$0.36650 \le p^\star \le 0.39775 \text{ and } 0.0625 \le p \le 0.03125.$$

for all the three levels of security.
Of course the sets $\mathcal{N}$ and $\mathcal{N}^\star$ are easily distinguishable from each other given sample of signatures with sufficient size since each set has different probability. However, the set $\mathcal{T}_{\mathbf{e}^\star}$, which is subset of $\mathcal{N}^\star$, is statistically indistinguishable from $\mathcal{N}^\star$.

4.4 Algorithm Sign in Game Notation

Algorithm Sign may be expressed more elegantly in game notation

```
Sign$(m, sk^\star)$
    $\mathbf{h} \leftarrow \mathcal{H}(m)$
    $\mathbf{e} = (e_1, \ldots, e_n) \leftarrow U\mathbf{h}$
    $\mathcal{J} \leftarrow \{c \mid c = (i-1)2^\epsilon + \sum_{k=1}^{\epsilon} e_{s_{ik}} 2^{k-1}, 1 \le i \le \theta\}$        (21)
    $\boldsymbol{\sigma} \leftarrow \sum_{c\in\mathcal{J}} G_c + \mathbf{e}$
    $\varsigma \leftarrow \sum_{c\in\mathcal{J}} T_c$
    return $(m, \boldsymbol{\sigma}, \varsigma)$
```

5 Security of the Scheme

5.1 Security notion

Attack Model We use the model of *Existential Unforgeability Under Chosen Message Attacks* to define security of the scheme. In this model [9], the adversary is given an oracle access to the signing algorithm, so that he can interact adaptively with it, issuing and receiving polynomially bounded number q of signing queries, and eventually coming up with a new message m that has not been signed before and trying to forge a valid signature for it.

Adversarial Model Let $\mathcal{A}$ be probabilistic polynomial-time adversary. The security definition is given by experiment EF–CMA as follows.

```
Expriment : EF–CMA($\mathcal{A}$)
    $(sk^\star, pk^\star) \leftarrow \mathsf{KeyGen}^\star(n, k, d, L, \epsilon)$
    $(m, \boldsymbol{\sigma}, \varsigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot, sk^\star)}(pk^\star)$
    return $(m, \boldsymbol{\sigma}, \varsigma)$
Sign($m, sk^\star$)
    $\mathbf{h} \leftarrow \mathcal{H}(m)$
    $\mathbf{e} = (e_1, \dots, e_n) \leftarrow U\mathbf{h}$
    $\mathcal{J} \leftarrow \{c \mid c = (i-1)2^\epsilon + \sum_{k=1}^{\epsilon} e_{s_{ik}} 2^{k-1}, 1 \le i \le \theta\}$
    $\boldsymbol{\sigma} \leftarrow \sum_{c\in\mathcal{J}} G_c + \mathbf{e}$
    $\varsigma \leftarrow \sum_{c\in\mathcal{J}} T_c$
    return $(m, \boldsymbol{\sigma}, \varsigma)$
```

Definition 3. *We define an advantage function of the adversary $\mathcal{A}$ as*

$$Adv^{EF-CMA}(\mathcal{A}) = Pr\left[H^\star \boldsymbol{\sigma} = \mathbf{s}^\star \text{ and } wt(\boldsymbol{\sigma}) = d\right],$$

where $\mathbf{s}^\star = \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k$, $\mathbf{s} = R\mathbf{h}$ and $\mathbf{h} = \mathcal{H}(m)$, and the message m is a new message that has not been signed by the Algorithm $\mathsf{Sign}(\cdot, sk)$.

It is obvious that in spite of the fact the adversary is issuing signing queries the for the messages $m_1, \dots, m_q$ adaptively, yet the corresponding hash values

$$\mathcal{H}(m_1), \dots, \mathcal{H}(m_q),$$

where $\mathcal{H}(m_i) \in \{0,1\}^L$, are produced randomly (out of the adversary's control). Consequently, for every arbitrary message m_i, the vector $\mathbf{s}_i = R\mathbf{h}_i$, where $\mathbf{h}_i = \mathcal{H}(m_i)$, is random vector out of the adversary's control. Thus, the adaptive interaction of the adversary $\mathcal{A}$, which consists of the signing queries from $\mathcal{A}$ and the corresponding signatures generated by the algorithm $\mathsf{Sign}(\cdot, sk^\star)$, can be captured by the distribution of the random triplets

$$\Pi_{\mathbf{s},\boldsymbol{\sigma},\varsigma,q} = \{(\mathbf{s}_1, \boldsymbol{\sigma}_1, \varsigma_1), \dots, (\mathbf{s}_q, \boldsymbol{\sigma}_q, \varsigma_q)\}. \tag{22}$$

5.2 Existential Unforgeability Under Chosen Message Attacks

The existential forgery can be defined with following problem.

*Problem 1 (*EF – CMA *Problem).* Given the public key $pk^\star = (H^\star, R, \mathcal{H}, d)$ with $H^\star \in \mathbb{F}_n^{(n-k)\times n}$, $R \in \mathbb{F}_2^{(n-k)\times L}$ along with the distribution

$$\Pi_{\mathbf{s},\boldsymbol{\sigma},\varsigma,q} = \{(\mathbf{s}_1, \boldsymbol{\sigma}_1, \varsigma_1), \dots, (\mathbf{s}_q, \boldsymbol{\sigma}_q, \varsigma_q)\},$$

where $(\mathbf{s}_i, \boldsymbol{\sigma}_i, \varsigma_i) \in \text{Col } R \times \mathbb{F}_2^n \times \mathbb{F}_2^\theta$, where Col R is the column span of the matrix R, such that

$$H^\star \boldsymbol{\sigma}_i = \mathbf{s}_i + \sum_{k=1}^{\theta} R_k \varsigma_{ik} \text{ and } \mathrm{wt}(\boldsymbol{\sigma}_i) = d, \text{ for } 1 \le i \le q.$$

Then given a random vector $\mathbf{s} \in \text{Col } R$, find a pair of vectors $(\boldsymbol{\sigma}, \boldsymbol{\varsigma}) \in \mathbb{F}_2^n \times \mathbb{F}_2^\theta$ satisfying

$$H^\star \boldsymbol{\sigma} = \mathbf{s}^\star \text{ and } \text{wt}(\boldsymbol{\sigma}) = d, \tag{23}$$

where $\mathbf{s}^\star = \mathbf{s} + \sum_{k=1}^{\theta} R_k \varsigma_k$.

Coding Theory Let $\mathcal{C}$ denote an $[n, k, d]$ random binary linear code with bock length n, dimension k, and distance d, where $d = n\mathsf{H}^{-1}(1 - \frac{k}{n})$ and H^{-1} is inverse of the binary entropy function $\mathsf{H}(x) = -x\log(x) - (1-x)\log(1-x)$.

Conjecture 1. We assume that it is computationally hard to distinguish the matrix $H^\star$ from a parity check matrix of an $[n, k, d]$ random binary linear code.

Justification. As we see from Algorithm $\mathsf{KeyGen}^\star$ 6, the matrix $H^\star$ is derived from the secret random matrix H by replacing random l columns from H with linear combinations of random columns from the matrices H and R and one column from the secret set $\{\mathbf{a}_1, \ldots, \mathbf{a}_\theta\}$ (see Equation (12)). In particular, for every $c = (i-1)2^\epsilon + j$, for $1 \le i \le \theta$, $0 \le j \le 2^\epsilon - 1$, we have

$$\sum_{k \in \mathcal{I}_c} H_k^\star + \sum_{k=0}^{\theta} R_k T_{k,c} = \mathbf{a}_i,$$

where $\mathcal{I}_c$ is random secret set with size $|\mathcal{I}_c|$ such that $\tau_i \le |\mathcal{I}_c| \le \tau_i + \epsilon$, $T \in \mathbb{F}_2^{\theta \times l}$ is an independent random secret matrix and the set $\{\mathbf{a}_1, \ldots, \mathbf{a}_\theta\}$ is a secret set of random vectors satisfying $\sum_{i=1}^{\theta} \mathbf{a}_i = \mathbf{0}$. So, the only difference between the matrix $H^\star$ and the random is that $H^\star$ contains these linear dependencies between some of its columns, the first θ columns of the matrix R, and secret set of vectors $\{\mathbf{a}_1, \ldots, \mathbf{a}_\theta\}$. Precisely, $\sum_{k \in \mathcal{I}_c} H_k^\star + \sum_{k=0}^{\theta} R_k T_{k,c} = \mathbf{a}_i$. Since the set $\mathcal{I}_c$, its size $|\mathcal{I}_c|$, the matrix T and (most importantly) the vector $\mathbf{a_i}$ are all secret, we assume that it is hard to find these linear combinations of columns that add up to the unknown vector $\mathbf{a}_i$ and hence it is computationally hard to distinguish $H^\star$ from the random.

Next, suppose that our security parameters n and k are chosen such that the syndrome decoding instance (n, k, d) is computationally hard for any random matrix $H \in \mathbb{F}_2^{(n-k)\times n}$, random vector (i.e., syndrome) $\mathbf{s}^\star \in \mathbb{F}_2^{n-k}$. Then, it is computationally hard to solve the $\mathsf{EF-CMA}$ problem by assuming that $H^\star$ is random $(n-k) \times n$ matrix, because doing so would imply solving a syndrome decoding instance (n, k, d), which is computationally hard for the selected n, k, and d. In other words, it is computationally hard to solve the $\mathsf{EF-CMA}$ problem by regarding the matrix $H^\star$ as a parity-check matrix for a random linear code $\mathcal{C}$.

We conjecture that no efficient way to solve Equation (23) for the pair of vectors $(\boldsymbol{\sigma}$,$\boldsymbol{\varsigma})$ other than using the syndrome decoding algorithms which deal with $H^\star$ as a random parity check matrix for a random linear code and with $\mathbf{s}^\star$ as a syndrome for some erroneous codeword. Therefore, if it is hard to solve Equation (23) as a syndrome decoding instance, then the scheme is secure against existential forgery. That is, it is computationally hard to a forge valid signature by solving the equation $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$ for the unknown $\boldsymbol{\sigma}$ such that $\mathrm{wt}(\boldsymbol{\sigma}) = d$.

6 Security Parameters

The syndrome decoding instance (n, k, d) can be made arbitrarily hard by increasing n and k to the required level of complexity. So, we chose our security parameters such that the time complexity of the instance satisfies the required level of security. Note that we are not interested in an error correction in this context; we only use the parameters n, k and d to estimate hardness of the relevant decoding problem. To this end, we use a *Syndrome Decoding Estimator* developed by Andre Esser and Emanuele Bellini which provides us with with concrete estimations for time and memory complexities for the best known algorithms for syndrome decoding. Then we choose our security parameters accordingly. The parameters sets are shown in Table 2.

6.1 Syndrome Decoding Estimator

In [8] Andre Esser and Emanuele Bellini give concrete hardness estimations for complexities of the best known solving algorithms for the syndrome decoding problem. In their paper, which was initially motivated by the need for determining the secure parameter sets for the code-based schemes in NIST's standardization process for post-quantum cryptography, Esser and Bellini developed a framework that allows them to obtain several of the major Information Set Decoding algorithms. Then, these algorithms are analyzed to drive formulas that give the concrete complexity of solving the syndrome decoding problem for each algorithm. Furthermore, they have implemented their framework into a software called *Syndrome Decoding Estimator* which is made available online [2].
We use Esser-Bellini's syndrome decoding estimator to estimate hardness of the syndrome decoding instance (n, k, d) for the $[n, k, d]$-code $\mathcal{C}$. Table 4 shows bit complexity estimation of several Information Set Decoding algorithms for solving the syndrome decoding instance with Parameters (n, k, d).

Following caution of Esser and Bellini we should mention that these estimations serve mainly as indicative benchmarks for ranges of parameter values and the corresponding sizes of the keys and signatures for the required level of security. For more details about the syndrome decoding estimator see the paper [8].

7 Conclusion and Further Research

We proposed a simple self-contained digital signature scheme that is easy to investigate and analyze. We also proposed concrete security parameters sets for

[2] `https://github.com/Crypto-TII/syndrome_decoding_estimator`

Table 4. Bit Complexity Estimation of Several Information Set Decoding Algorithms for Solving the Syndrome Decoding Instance with Parameters (n, k, d) using Esser-Bellini's Estimator.

(n,k,d)	$(1600, 400, 343)$		$(2432, 608, 521)$		$(3264, 816, 700)$	
Algorithm	Time	Memory	Time	Memory	Time	Memory
Prange [12]	192.5	21.2	279.1	22.3	365.7	23.2
Stern [14]	172.9	42.9	256.4	62.8	340.3	82.6
Dumer [7]	172.6	48.7	256.1	58.3	341.5	49.4
Ball Collision [4]	172.9	42.9	256.4	62.8	340.2	87.8
BJMM (MMT) [10]	167.5	94.3	245.0	159.5	323.5	186.5
BJMM-pdw [3]	169.3	72.8	248.7	107.6	329.1	115.4
May-Ozerov [11]	167.0	106.5	246.4	112.6	327.8	115.0
Both-May [5]	170.7	66.0	251.5	77.7	333.8	78.6

k-bit security ($k = 128, 192, 256$) with justification based on known attacks. The scheme has neat and efficient signing and verifying algorithms and very small signatures sizes.
Although the scheme has large public key, in practice this problem may be addressed. Note that the verification equation has the form $H^{\star}\boldsymbol{\sigma} = \mathbf{s}^{\star}$. The verifying party may choose random rows from the matrix $H^{\star}$ and their corresponding rows of the matrix R to perform the verification. Thus, the verifying party can secretly choose and store *a small random part of the public key* and use it for the verification instead of the whole public key. Hence the verifying party needs to store only very small part of the key, *a small secret verification key* chosen randomly from the public key. In this way we can drastically reduce the size of the *verification key* which stored on the verifying party side. However, this part must be kept secretly. This mode of operation improves performance of the verification in terms of both memory and run-time without affecting the security, since the adversary does not know which part of the public key will be used for the verification..
Another possible improvements are structures of the matrices $H^{\star}$ and G. We have seen that randomness of the matrix $H^{\star}$ is crucial for security of the scheme. There are many other possible ways for modifying the matrix H of the basic scheme into $H^{\star}$ and simultaneously creating the corresponding matrix G. The goal here is to make $H^{\star}$ as indistinguishable from the random as possible while keeping the correspondence between the verification and the syndrome decoding instance unchanged.
However, better randomization may require some slight modifications in the verification algorithm. For a better randomization of the matrix $H^{\star}$ we may weaken the restriction on the weight of the vector $\boldsymbol{\sigma}$ to make the weigh lie within certain range within which the underlying problem is guaranteed to remain computationally hard; that is we restrict $\mathrm{wt}(\boldsymbol{\sigma})$ such that $d - \delta \le \mathrm{wt}(\boldsymbol{\sigma}) \le d + \delta$ instead of $\mathrm{wt}(\boldsymbol{\sigma}) = d$ and hence we gain more freedom in randomizing the matrices $H^{\star}$ and G as well as the signature. Also, another possible randomization may

be achieved by changing the equality $H^\star \boldsymbol{\sigma} = \mathbf{s}^\star$ to a proximity, for example $H^\star \boldsymbol{\sigma} \approx_\delta \mathbf{s}^\star$ or $d(H^\star \boldsymbol{\sigma}, \mathbf{s}^\star) \leq \delta$.
Any of these modifications come at cost of more complexity and require careful investigation and analysis.

References

1. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: A finite regime analysis of information set decoding algorithms. Algorithms **12**(10), 209 (2019)
2. Baldi, M., Bitzer, S., Pavoni, A., Santini, P., Wachter-Zeh, A., Weger, V.: Zero knowledge protocols and signatures from the restricted syndrome decoding problem. Cryptology ePrint Archive, Paper 2023/385 (2023), `https://eprint.iacr.org/2023/385`, `https://eprint.iacr.org/2023/385`
3. Becker, A., Joux, A., May, A., Meurer, A.: Decoding random binary linear codes in $2^{n/20}$: How 1+ 1= 0 improves information set decoding. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 520–536. Springer (2012)
4. Bernstein, D.J., Lange, T., Peters, C.: Smaller decoding exponents: ball-collision decoding. In: Annual Cryptology Conference. pp. 743–760. Springer (2011)
5. Both, L., May, A.: Decoding linear codes with high error rate and its impact for lpn security. In: International Conference on Post-Quantum Cryptography. pp. 25–46. Springer (2018)
6. Courtois, N.T., Finiasz, M., Sendrier, N.: How to achieve a mceliece-based digital signature scheme. In: Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. pp. 157–174. Springer (2001)
7. Dumer, I.: On minimum distance decoding of linear codes. In: Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory. pp. 50–52 (1991)
8. Esser, A., Bellini, E.: Syndrome decoding estimator. In: IACR International Conference on Public-Key Cryptography. pp. 112–141. Springer (2022)
9. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on computing **17**(2), 281–308 (1988)
10. May, A., Meurer, A., Thomae, E.: Decoding random linear codes in $\mathcal{O}(2^{0.054n})$. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 107–124. Springer (2011)
11. May, A., Ozerov, I.: On computing nearest neighbors with applications to decoding of binary linear codes. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 203–228. Springer (2015)
12. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory **8**(5), 5–9 (1962)
13. Ren, F., Zheng, D., Wang, W., et al.: An efficient code based digital signature algorithm. Int. J. Netw. Secur. **19**(6), 1072–1079 (2017)
14. Stern, J.: A method for finding codewords of small weight. In: International colloquium on coding theory and applications. pp. 106–113. Springer (1988)