Digital Signature from Syndrome Decoding Problem

Abdelhaliem Babiker
Dept. of Basic Eng. Sciences, College of Engineering
Imam Abdulrahman Bin Faisal University
Dammam, Saudi Arabia
aababiker@iau.edu.sa

***Abstract*—This paper introduces new digital signature scheme whose security against existential forgery under adaptive chosen message attack is based on hardness of the Syndrome Decoding Problem. The hardness assumption is quite simple and hence easy to analyze and investigate. The scheme as whole is neat with intuitive security definition and proof in addition to elegant and efficient signing and verifying algorithms. We propose parameter sets for three security levels (128-bits, 192-bits, and 256 bits) and estimate the corresponding sizes of the keys and the signature for each level. Additionally, the scheme has an interesting feature of signature verification using an arbitrary part of the public key, which allows the verifying party to store a small random *secret* part of the public key rather than the full-size public key. Using small part of the public key for verification gives us more time and memory efficient verification mode which we call *Light Verification Key Mode* (LVK) mode. Also, we suggest *Light Signing Key Mode* (LSK) which enables a smaller size of the private (signing) key while maintaining the same security level.**

I. Introduction

Many public-key cryptosystems are under threat of the large-scale quantum computers which are anticipated to be built in the foreseeable future. In particular, cryptosystems that are based on the Discrete Logarithm Problem (DLP) or the Integer Factorization Problem will be compromised using Shor's algorithms [1] which are designed to exploit quantum computers to solve these problems. Therefore, there has recently been great interest in cryptosystems that are based on hardness of different problems which are assumed to resist attacks of both classical and quantum algorithms. One of these problems is *Syndrome Decoding Problem* which has long history of study and forms basis for security of many cryptosystems, for example [2]–[4]. The syndrome decoding problem is closely connected to *Learning Parity with Noise* problem which also has many cryptographic applications [5].

In response to the ongoing advancements of quantum computing, National Institute of Standards and Technology (NIST) has initiated in 2016 a competition-like standardization process for selecting quantum-safe public-key algorithms to be standardized in order to replace the currently used public-key algorithms. In July 2022, after three rounds, NIST announced selection of one key encapsulation mechanism and three digital signature algorithms to be standardized and additional four public-key encryption and key-establishment algorithms for the fourth-round of standardization process. Three of these fourth-round finalists, namely Classic McEliece [2], BIKE [3] and HQC [4] are based on the hardness of the syndrome decoding problem. Except for one algorithm, all of the selected algorithms for standardization are lattice-based algorithms. As there is no fourth-round selected digital signature algorithm, NIST is calling for additional digital signature proposals [6], which are preferred to be based on hardness assumptions other than structured lattices which form basis for three out of the four selected algorithms.

This paper introduces *Signature from Syndrome Decoding* (SSD, for short), a new digital signature scheme that is based on hardness of syndrome decoding problem. More precisely, we formulate definition of security of the signature against existential forgery under adaptive chosen message attack into an assumption which is based on the hardness of the syndrome decoding problem.

Paper Organization: We start in Section 2 with high level description of the scheme in order to give the reader the big picture before we delve into the details in Section 3 in which we devise our basic building blocks. Next, we introduce the main hardness assumption in Section 4 and give justification for its hardness. Thence we present the signature scheme in Section 5 and prove its correctness. Security analysis of the scheme is given in Section 6. Also, in this section we define the security levels of the scheme and the corresponding parameter sets. In Section 7 we give estimations of hardness of the underlying syndrome decoding instances of the parameter sets of the scheme. Finally, we devote Section 8 and to the Light Verification Key (LVK) mode of the scheme along with its security estimation, and Section 9 for brief description of the Light Signing Key mode (LSK).

II. Digital Signature Scheme: High Level Description

Suppose that we have a message m which we want to sign using our digital signature scheme. First we specify the pair of the private and public key which are used for signing and verifying, respectively. Thence we show how to use these keys to sign and verify a signature, and why the scheme should be secure, i.e. why it is hard to forge a signature for any arbitrary message. Also, we explain the rationale behind the way in which the scheme is designed. The details of how the scheme works are given in the subsequent sections. The reader may also come back to this section after completing reading the paper. In this case this section provides the summary of the scheme's description.

1) The Keys: Let us start with publicly known random matrix $Y \in \mathbb{F}_2^{k' \times k}$ and three secret random matrices $S \in \mathbb{F}_2^{k \times n}$, $K \in \mathbb{F}_2^{n \times n}$, and $P \in \mathbb{F}_2^{n \times n}$, where P is permutation matrix. All of these four matrices are randomly constructed but with specific structure in order to enable the required functionality of the scheme. Then we set $H \leftarrow YSP^{-1}$. Next, we keep the matrices K and P privately, and we set as a public key the matrices Y and H in addition to some hash function $\mathcal{H}$ such that

$$\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{F}_2^n.$$

In other words, we set

$$pk = (Y, H, \mathcal{H}) \text{ and } sk = (K, P).$$

We conjecture and prove that the matrix Y is computationally indistinguishable from the random and that it is hard to recover the secret matrices S and P from H and Y (Theorem 3).

2) Signing: Now, to sign the message m, we obtain

$$\mathbf{h} \leftarrow \mathcal{H}(m).$$

That is, $\mathbf{h} \in \mathbb{F}_2^n$ is hash value of the message m. Then we generate the signature as

$$\mathbf{s} \leftarrow P(K\mathbf{h} \oplus \mathbf{r}),$$

where $\mathbf{r} \in \mathbb{F}_2^n$ is a secret random vector obtained from some random subspace of $\mathbb{F}_2^n$. In fact, the vector $\mathbf{r}$ represents *random noise* added to the vector $K\mathbf{h}$ before applying the secret permutation on the sum $(K\mathbf{h} \oplus \mathbf{r})$. All of the permutation matrix P, the matrix K, and the noise vector $\mathbf{r}$ are secret, and for every new signature, a new random independent noise vector $\mathbf{r}$ is used. Thus, we make sure that the signatures do not reveal the secret matrices K and P.

3) Verifying: The pair $(m, \mathbf{s})$ of the message and its signature is verified as follows. First the receiver compute the hash value of m, $\mathbf{h} \leftarrow \mathcal{H}(m)$, where $\mathbf{h} = [\ \mathbf{h}_1 \quad \cdots \quad \mathbf{h}_\gamma\ ]^T$ and $\mathbf{h}_i \in \mathbb{F}_2^k$, for $1 \leq i \leq \gamma$. That is, the hash value is viewed as γ equal-length parts for some nonegative integer γ and each part is represented as a vector in $\mathbb{F}_2^k$. Then the receiver checks if

$$H\mathbf{s} = Y\bar{\mathbf{h}}, \text{ where } \bar{\mathbf{h}} = \sum_{j=1}^{\gamma} \mathbf{h}_i.$$

The receiver accepts the signature if and only if the equality holds and the Hamming weight of the vector $\mathbf{s}$ is within certain range.

A. Performance of the Scheme

It is easy to see that the scheme efficient with signing and verifying preformed in $O(n^2)$ operations. Size of each of the the public key and private key is of order $O(n^2)$ and the signature size is n bits.

B. Security of the Scheme

The verification equation $H\mathbf{s} = Y\bar{\mathbf{h}}$ corresponds to an instance of syndrome decoding problem in which rows of H are subset of rows of some random parity check matrix for some (unknown) random binary linear $[n, k', d]$-code $\mathcal{C}$, the vector $(\mathbf{v} = Y\bar{\mathbf{h}})$ represents syndrome of some (unknown) codeword, and the signature $\mathbf{s}$ is a solution for this instance. (See Section 4 for full details.)

Therefore, if it is hard to solve the syndrome decoding instance $(H, \omega, \mathbf{v})$ for $\mathbf{s}$, where ω is the Hamming weight of $\mathbf{s}$, then given the message m, it is hard to forge a signature $\mathbf{s}'$ with weight $wt(\mathbf{s}') \approx \omega$ such that $H\mathbf{s}' = Y\bar{\mathbf{h}}$.

1) Hamming Weight of the Signature: Hardness of syndrome decoding instance $(H, \omega, \mathbf{v})$ depends on the parameter ω as well as dimensions of the matrix H (i.e., k' and n). We choose ω, which represents the Hamming weight of the legitimate signature in our scheme, to be restricted within certain rage. Particularly, the upper bound on ω is restricted by Gilbert–Varshamov bound of d and $\frac{k}{n}$. The lower bound on ω can be made arbitrarily close to the upper bound.

To achieve this restriction on weight of the signature we designed the signature algorithm such that the sum after adding the random noise vector $\mathbf{r}$ to $K\mathbf{h}$ results in the required weight. The whole operation can be randomized such that the random vector $\mathbf{r}$ in the sum $(K\mathbf{h} \oplus \mathbf{r})$ is uniformly drown from random subspace of $\mathbb{F}_2^n$ which we call the *noise distribution*. Note that weight of the signature $\mathbf{s} = P(K\mathbf{h} \oplus \mathbf{r})$ is same as the weight of the vector $(K\mathbf{h} \oplus \mathbf{r})$, since multiplication by the permutation matrix P preserves the weight.

2) Parameter Sets and Security Levels: We choose the parameters n and k' such that the syndrome decoding instance $(H, \omega, \mathbf{v})$ is sufficiently hard. To do this, we use estimation of Andre Esser and Emanuele Bellini [7] for hardness of the syndrome decoding problem, which is based on the performance of the best known information set decoding algorithms for solving the syndrome decoding problem. Thus, we determine the parameter sets and their corresponding security level using Esser-Bellini estimations as reference.

We provide thee parameter sets for three security levels (128-bits, 192-bits, and 256-bits). Table I shows these security levels along with the parameters of the underlying syndrome decoding instance and sizes of the keys and the signature.

C. Light Verification Key (LVK) Mode

For efficiency in terms of both time and space required by the verification algorithm, we introduce mode of operation which we call *Light Verification Key Mode*. In this mode, the receiver (or verifying party) keeps only a small part of the public key rather than storing the entire public key, and he uses this small part for verifying the signature. However, this part must be kept privately. In other words, the receiver keeps a private verification key which is derived randomly from the original public key as follows. From public key

$$pk = (H, Y, \mathcal{H})$$

TABLE I
SECURITY LEVELS, THE UNDERLYING SYNDROME DECODING INSTANCE (PARAMETER SETS) AND, SIZES OF KEYS AND THE SIGNATURE FOR THE STANDARD *SSD*.

Security Level	128-bits	192-bits	256-bits
Parameters (n, k', ω)	$(1760, 242, 484)$	$(2744, 377, 754)$	$(3728, 512, 1025)$
Public Key Size (KB)	65	158	291.5
Private Key Size (KB)	27	63	113
Signature Size (Bytes)	220	343	466

KB = 1024 Bytes.

the verifying party secretly forms the matrices H' and Y' from rows of H and Y, respectively, by selecting random k'' rows from the matrix H and k'' of their corresponding counterparts from Y, where $k'' < k'$. Then the verifying party forms the new *light verification key*

$$pk^{'} = (H', Y', \mathcal{H}).$$

Signature verification is performed in the same way as in the standard mode; the verifying party accepts the signature $\mathbf{s}$ if and only if weight of $\mathbf{s}$ is within the legitimate range and

$$H'\mathbf{s} = Y'\bar{\mathbf{h}}.$$

Note that there are $\binom{k'}{k''}$ way for driving the secret verification key form the public key. We determine three parameter sets for the three security levels (128-bits, 192-bits, and 256-bits) using estimation of hardness of the underlying syndrome decoding problem $(H', \omega, Y'\bar{\mathbf{h}})$, which is significantly easier compared to syndrome decoding instance associated with the full-size public key, since k'' is less than k', but however the adversary does know which instance to solve among the $\binom{k'}{k''}$ possible instances. Table II shows security levels for the light verification key mode (LVK) in which the public key is replaced with a secret verification key. Wee see from the table

TABLE II
SIZES OF THE KEYS AND THE SIGNATURE FOR DIFFERENT SECURITY LEVELS OF THE LVK-*SSD*.

Security Level	128-bits	192-bits	256-bits
Parameters (n, k'', ω)	$(1760, 32, 484)$	$(2744, 40, 754)$	$(3728, 48, 1025)$
LVK Size (KB)	9	17	27.5
LSK Size (KB)	4.5	7.5	10.5
(Standard) Private Key Size (KB)	27	63	113
Signature Size (Bytes)	220	343	466

that size of the verification key is reduced more than seven times for the 128-bits security level, nine times for the 192-bits security level, and ten times for the 256-bits security level. Also the verification runtime is reduced since rows of matrices H' and Y' are small fractions of rows of the matrices H and Y. For example in the 192-bit security level H is 377×2744 matrix, while H' is only 40×2744.

D. Light Signing Key (LSK) Mode

Size of private key $sk = (K, P)$ also can be reduced by specifying spacial structure for the matrix K which allows K to be stored and manipulated efficiently. We call this mode of operation *Light Signing Key (LSK) Mode*. Dimensions of the matrices K and P do not change, only structure of the matrix K changes. Unlike the LVK mode which does not require change in the key generation algorithms, in LSK mode structure of both public and private keys change.
Table II shows sizes of the private key (i.e. signing key) in both standard mode and LSK mode in which the standard signing key is replaced with a space-efficient light signing key (LSK). We see that LSK mod has more than six times smaller signing key size for 128-bit security level, nine times smaller signing key size for 192-bit security level, and more than eleven times smaller signing key size for 256-bit security level. Note that, as Table II suggests, LVK can also be combined with LSK mode.

III. PRELIMINARIES

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 arithmetic is carried over $\mathbb{F}_2$, and matrix addition is referred to by bitwise xor, $\oplus$.

1) Notation: $s \xleftarrow{R} S$ means element s is uniformly at random selected from the set S. We use $|S|$ to denote size of the set S.
Let $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_\gamma \end{bmatrix}$, where $\mathbf{x}_i \in \mathbb{F}_2^k$, for $1 \leq i \leq \gamma$. We define $\bar{\mathbf{x}}$ as $\bar{\mathbf{x}} = \sum_{i=1}^{\gamma} \mathbf{x}_i$. We use $wt(\mathbf{v})$ to denote the Hamming weight of the binary vector $\mathbf{v}$, that is the number of the 1s in $\mathbf{v}$.

Next, let M be nonsingular matrix in $\mathbb{F}_2^{k\times k}$ with multiplicative order $\phi = 2r$, for some odd number $r \in \mathbb{N}$. In particular, we define M as

$$M = F^{-1}GF, \tag{1}$$

where $F \in \mathbb{F}_2^{k\times k}$ is uniformly random full-rank matrix and $G \in \mathbb{F}_2^{k\times k}$, such that

$$G = \begin{bmatrix} D & \\ & C \end{bmatrix} \text{ with } D^2 = I_l \text{ and } C^r = I_m,$$

where D is $l \times l$ matrix, C is $m \times m$ matrix, and I_l, I_m are $l \times l$, $m \times m$ identity matrices, respectively. Thus, the matrix M has multiplicative order of $\phi = 2r$. That is,

$$M^\phi = F^{-1}\begin{bmatrix} D^{2r} & \\ & C^{2r} \end{bmatrix} F = F^{-1}\begin{bmatrix} I_l & \\ & I_m \end{bmatrix} F = I_k.$$

A. Construction of the Matrix M

It is easy to construct a matrix M as defined in (1). One way to do that is as follows.

The Matrix F: the matrix $F \in \mathbb{F}_2^{k\times k}$ is full-rank matrix which can easily be generated randomly.

The Matrix D: the matrix D can be any random $l \times l$ involutory matrix. However. for simplicity of presentation and analysis, and also for minimizing memory consumption and computational cost, we choose D to be direct sum of random matrices D_i, for $0 \le i \le \frac{l}{2}$, where

$$D_i \in \left\{ \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right\}.$$

That is,

$$D = \begin{bmatrix} D_1 & & \\ & \ddots & \\ & & D_{l/2} \end{bmatrix}.$$

The matrices D_i will be chosen uniformly at random, and thus there are $2^{\frac{l}{2}}$ possible combinations for the matrix D. Furthermore, there are many other possible ways for generating random involutory matrix (see for example [8] and [9]).

The Matrix C: we construct the matrix C such that C has multiplicative order r. It can be constructed as

$$C = \begin{bmatrix} C_1 & & \\ & \ddots & \\ & & C_t \end{bmatrix},$$

where each C_i, for $1 \le i \le t$, is a companion matrix of which a minimal polynomial $p_i(x) \in \mathbb{F}_2[x]$ has multiplicative order r_i, where r_i is odd. Thus, the multiplicative order of C is the least common multiple of r_1,..., r_t. That is, $r = lcm(r_1, ..., r_t)$.

The matrix M, which can easily be generated randomly, has the following property.

Let $Q = (I_k \oplus M^r)$. Then,

$$M^\theta Q = QM^\theta = Q, \text{ for every integer } \theta. \tag{2}$$

Proof: It is obvious that $QM = MQ$, since $Q = I_k \oplus M^r$. Next, notice that

$$Q = (I_k \oplus M^r) = F(I_k \oplus G^r)F^{-1} = F \begin{bmatrix} I_l \oplus D & \\ & 0_{m\times m} \end{bmatrix} F^{-1}.$$

On the other hand

$$\begin{aligned} QM &= (I_k \oplus M^r)M \\ &= (I_k \oplus FG^rF^{-1})FGF^{-1} \\ &= F(I_k \oplus G^r)F^{-1}FGF^{-1} \\ &= F(I_k \oplus G^r)GF^{-1}. \end{aligned}$$

Since $I_k \oplus G^r = \begin{bmatrix} I_l \oplus D & \\ & 0_{m\times m} \end{bmatrix}$ and $G = \begin{bmatrix} D & \\ & C \end{bmatrix}$,

$$\begin{aligned} QM &= F\left(I_k \oplus G^r\right)GF^{-1} \\ &= F \begin{bmatrix} I_l \oplus D & \\ & 0_{m\times m} \end{bmatrix} \begin{bmatrix} D & \\ & C \end{bmatrix} F^{-1} \\ &= F \begin{bmatrix} (I_l \oplus D)D & \\ & 0_{m\times m}C \end{bmatrix} F^{-1}. \end{aligned}$$

We have $(I_l \oplus D)D = (D \oplus D^2) = (D \oplus I_l)$. Hence,

$$QM = F \begin{bmatrix} I_l \oplus D & \\ & 0_{m\times m} \end{bmatrix} F^{-1} = Q.$$

Using induction on θ we see that $QM^\theta = QMM^{\theta-1} = QM^{\theta-1} = ... = Q$. ∎

B. Column Space of the Matrix Q: Noise Distribution $\mathcal{N}^\gamma$

Consider

$$\begin{aligned} Q &= (I_k \oplus M^r) = F^{-1}(I_k \oplus G^r)F \\ &= F \begin{bmatrix} I_l \oplus D & \\ & 0_{m\times m} \end{bmatrix} F^{-1} \\ &= \begin{bmatrix} F \begin{bmatrix} b_{1,1} \\ \vdots \\ b_{l,1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \cdots & F \begin{bmatrix} b_{1,n} \\ \vdots \\ b_{l,n} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{bmatrix}, \end{aligned}$$

where $b_{i,j} = ((I_l \oplus D)F)_{i,j} \in \{0, 1\}$.

Let $l' = \frac{l}{2}$. Given structure of the matrix D, it is easy to see that the sub-matrix $(I_l \oplus D)$ has exactly $l' = \frac{l}{2}$ nonzero rows, and these nonzero rows are linearly independent. Therefore, the matrix $\begin{bmatrix} I_l \oplus D & \\ & 0_{m\times m} \end{bmatrix}$ has rank l'.

Note that the matrix F is full-rank matrix. Since multiplication by a full-rank square matrix preserves rank [1], it is not hard to see that

$$\text{rank } Q = l' \text{ and dim Nul } Q = k - l'.$$

Observe that columns of the matrix Q live in the l-dimensional subspace spanned (or generated) by the first l columns of the matrix F.

Let the set $\mathcal{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_{l'}\}$ be basis for Col Q, where Col Q is the set of all linear combinations of the columns of Q

1) Remark: Note that $Q^2 = QQ = 0_{k\times k}$. Thus, Col $Q \subseteq$ Nul Q.

Now, let $\mathcal{N} =$ Col Q. We define the *noise distribution* $\mathcal{N}^\gamma$ associated with the matrix M as follows

$$\mathcal{N}^\gamma = \{\mathbf{e} \in \mathbb{F}_2^n : \mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_\gamma \end{bmatrix}, \text{where } \mathbf{e}_i \in \mathcal{N}, \text{ for } 1 \le i \le \gamma\}.$$

[1] https://statlect.com/matrix-algebra/matrix-product-and-rank

Note that $\mathcal{N}$ is l'-dimensional subspace of $\mathbb{F}_2^k$ with size $|\mathcal{N}| = 2^{l'}$. Therefore, size of the set $\mathcal{N}^\gamma$ which is subspace of $\mathbb{F}_2^n$ is

$$|\mathcal{N}^\gamma| = 2^{l'\gamma}.$$

The noise distribution has the following property. For every $\mathbf{e}_i \in \mathcal{N}$ we have

$$Q\mathbf{e}_i = \mathbf{0} \text{ and } M\mathbf{e}_i = \mathbf{e}_i. \tag{3}$$

Sine $\mathbf{e}_i \in (\mathcal{N} = \text{Col } Q)$ and Col $Q \subseteq$ Nul Q, it follows that $\mathbf{e}_i \in$ Nul Q and hence $Q\mathbf{e}_i = \mathbf{0}$. Also, since $MQ = Q$ and $\mathbf{e}_i \in$ Col Q, we get $M\mathbf{e}_i = \mathbf{e}_i$. Furthermore, since $M\mathbf{e}_i = \mathbf{e}_i$, it follows that $M^\alpha \mathbf{e}_i = M^{\alpha-1}M\mathbf{e}_i = M^{\alpha-1}\mathbf{e}_i = \cdots = \mathbf{e}_i$ for every integer α.

2) Purpose of the Noise Distribution $\mathcal{N}^\gamma$: We use the noise distribution to achieve three major goals; (a) providing randomness, thus (b) ensuring the secrecy of our underlying trapdoor structures, (c) also we utilize the noise in reducing the Hamming weight of the signature. In the public key the noise serves in hiding the underlying structure which characterizes the scheme and it virtue of which the verification works. Furthermore, we use the noise within the function $\mathcal{R}(\cdot)$ which reduces weight of the signature, and also this contributes in hiding both of the secret matrices K and P, as we will see.

C. Weight Reduction Function wrf

Consider the basis $\mathcal{B} = \{\mathbf{v}_1, \ldots, \mathbf{v}_{l'}\}$ of the subspace $\mathcal{N}$. Let $B = [\mathbf{v}_1 \cdots \mathbf{v}_{l'}]$. Consider the transpose matrix B^T, which is $l' \times k$ matrix. We apply Gaussian elimination on B^T to transform it into *reduced echelon form* $\tilde{B}^T$. Since the column vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{l'}$ are, by definition of the basis, linearly independent, the matrix B^T has l' linearly independent rows and its reduced echelon form $\tilde{B}^T$ has l' *pivot positions*.
Next, consider

$$\tilde{B} = (\tilde{B}^T)^T = [\tilde{\mathbf{v}}_1 \cdots \tilde{\mathbf{v}}_{l'}].$$

Let $\tilde{B}(:,t)$ be sub-matrix of $\tilde{B}$ containing random t columns of the matrix $\tilde{B}$ (assume that these columns are ordered according the pivot positions in $\tilde{B}^T$), where $t \leq l'$. Note that the transpose matrix $\tilde{B}(:,t)^T$ contains the corresponding t rows from the matrix $\tilde{B}^T$ and thus it has t pivot positions. Consider the set

$$\mathcal{P}_{\tilde{B}(:,t)^T} = \{(i,j) : (i,j) \text{ is pivot position in } \tilde{B}(:,t)^T\}.$$

That is, the set $\mathcal{P}_{\tilde{B}(:,t)^T}$ contains subscripts of the pivots in the sub-matrix $\tilde{B}(:,t)^T$ of the reduced echelon form matrix $\tilde{B}^T$.

Now, we define *Weight Reduction Function*

$$wrf(\cdot, \tilde{B}, t) : \mathbb{F}_2^k \longrightarrow \mathbb{F}_2^k$$

which reduces the Hamming weight of its input simply by setting some of its nonzero entries to zeros, as in the following algorithm. The function takes a vector $\mathbf{x} \in \mathbb{F}_2^k$ and ensures every j^{th} nonzero coordinate in $\mathbf{x}$ that corresponds to the j^{th} pivot column in $\tilde{B}(:,t)^T$ is turned into zero, for $1 \leq j \leq k$. Since the matrix $\tilde{B}(:,t)^T$ has t pivot columns, the function wrf ensures that the particular t coordinates of the vector $\mathbf{x}$ that correspond to the pivot positions in $\tilde{B}(:,t)^T$ are all set to zeros (of course some these coordinates are already zeros).

Algorithm 1 Weight Reduction Function wrf

Require: $\mathbf{x} \in \mathbb{F}_2^k$, $\tilde{B}, t$, such that $wt(\mathbf{x}) \approx \frac{k}{2}$ and $t \leq l'$.
Ensure: $\mathbf{x}' \in \mathbb{F}_2^k$ with weight $wt(\mathbf{x}') \approx \frac{k-t}{2}$. ▷ Let $(x_1, \ldots, x_k)$ be the binary representation of $\mathbf{x}$.

1:
2: **for** each $(i,j) \in \mathcal{P}_{\tilde{B}(:,t)^T}$ **do** ▷ i.e., for each pivot position (i,j) in $\tilde{B}(:,t)^T$.
3: **if** $x_j = 1$ **then**
4: $\mathbf{x} \leftarrow \mathbf{x} \oplus \tilde{\mathbf{v}}_i$; ▷ i.e., set $x_j \leftarrow 0$.
5: **end if**
6: **end for**
7: Output $\mathbf{x}$;

The function wrf can be defined more neatly as

$$wrf(\mathbf{x}, \tilde{B}, t) = \mathbf{x} \oplus \sum_{(i,j) \in \mathcal{P}_{\tilde{B}(:,t)^T}} x_j \tilde{\mathbf{v}}_i.$$

Note that for each $(i,j) \in \mathcal{P}_{\tilde{B}(:,t)^T}$, the vector $\tilde{\mathbf{v}}_i$ exists in the sum above if and only if $x_j = 1$. Since each $\tilde{\mathbf{v}}_i$ has 1 in the j^{th} position (because (i,j) is a pivot position in $\tilde{B}(:,t)^T$), the statement $\mathbf{x} \leftarrow \mathbf{x} \oplus \tilde{\mathbf{v}}_i$ turns the 1 in each corresponding j^{th} position in the vector $\mathbf{x}$ to 0. Thus, algorithm\function wrf ensures that the t entries of $\mathbf{x}$ that correspond the pivot columns in $\tilde{B}(:,t)^T$ are all zeros.

Theorem 1: Let $\mathbf{x}$ be uniformly random vector from $\mathbb{F}_2^k$ with weight $wt(\mathbf{x}) \approx \frac{k}{2}$. Then weight of $\mathbf{x}' = wrf(\mathbf{x}, \tilde{B}, t)$ equals $\frac{(k-t)}{2}$ (approximately). That is,

$$wt(\mathbf{x}') \approx \frac{(k-t)}{2}.$$

.

Proof: We have

$$\mathbf{x}' = wrf(\mathbf{x}, \tilde{B}, t) = \mathbf{x} \oplus \sum_{(i,j) \in \mathcal{P}_{\tilde{B}(:,t)^T}} x_j \tilde{\mathbf{v}}_i.$$

By definition of the wrf function, t entries of $\mathbf{x}'$ are zeros. Furthermore, since $\mathbf{x}$ is uniformly random, the remaining $(k-t)$ entries of $\mathbf{x}'$ will remain uniformly random with $\frac{(k-t)}{2}$ 1s (approximately). Therefore, $wt(\mathbf{x}') \approx \frac{(k-t)}{2}$. ■

1) Remark: Since $\mathcal{N} = \text{span}\{\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_{l'}\}$, it follows that

$$\left(\sum_{(i,j) \in \mathcal{P}_{\tilde{B}(:,t)^T}} x_j \tilde{\mathbf{v}}_i \right) \in \mathcal{N}.$$

Hence,

$$\mathbf{x}' = \mathbf{x} \oplus \mathbf{z},$$

for some $\mathbf{z} \in \mathcal{N}$, where $\mathbf{z} = \sum_{(i,j) \in \mathcal{P}_{\tilde{B}(:,t)^T}} x_j \tilde{\mathbf{v}}_i$.

Next, let $\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_\gamma \end{bmatrix}$ such that $\mathbf{u}_i \in \mathbb{F}_2^k$, $i \in [1, \gamma]$, where $k = \frac{n}{\gamma}$. We define the function

$$\mathcal{R}(\cdot, t) : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$$

as follows

$$\mathcal{R}(\mathbf{u}, t) = \begin{bmatrix} wrf(\mathbf{u}_1, \tilde{B}, t) \\ \vdots \\ wrf(\mathbf{u}_\gamma, \tilde{B}, t) \end{bmatrix}.$$

Assume that $\mathbf{u} \in \mathbb{F}_2^n$ is uniformly random vector with weight $wt(\mathbf{u}) \approx \frac{n}{2}$. Hence, each part $\mathbf{u}_i \in \mathbb{F}_2^k$ has weight $wt(\mathbf{u}_i) \approx \frac{k}{2}$.
Now, by Theorem 1, each $\mathbf{u}'_i = wrf(\mathbf{u}_i, \tilde{B}, t)$ has weight $wt(\mathbf{u}'_i) \approx \frac{(k-t)}{2}$. Therefore, weight of $\mathbf{u}' = \mathcal{R}(\mathbf{u}, t)$ is $wt(\mathbf{u}') = \sum_{i=1}^{\gamma} wt(\mathbf{u}'_i) \approx \frac{\gamma(k-t)}{2}$.

2) Remark: [Generalization of Remark III-C1] From Remark III-C1, each $\mathbf{u}'_i = wrf(\mathbf{u}_i, \tilde{B}, t)$ has the form $\mathbf{u}'_i = \mathbf{u}_i \oplus \mathbf{z}_i$, where $\mathbf{z}_i \in \mathcal{N}$. Therefore, the vector

$$\mathbf{u}' = \mathcal{R}(\mathbf{u}, t) = \begin{bmatrix} \mathbf{u}'_1 = \mathbf{u}_1 \oplus \mathbf{z}_1 \\ \vdots \\ \mathbf{u}'_\gamma = \mathbf{u}_\gamma \oplus \mathbf{z}_\gamma \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_\gamma \end{bmatrix} \oplus \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_\gamma \end{bmatrix}$$

has the form

$$\mathbf{u}' = \mathbf{u} \oplus \mathbf{r} \text{ with } \mathbf{r} \in \mathcal{N}^\gamma, \text{ where } \mathbf{r} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_\gamma \end{bmatrix}.$$

IV. Hardness Assumptions

A. Syndrome Decoding Problem

A random binary linear code $[n, k, d]-$code $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_2^n$ that can be defined as Nul space for a matrix $H \in \mathbb{F}_2^{(n-k)\times n}$. That is,

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

The matrix H is known as a *parity check matrix* for $\mathcal{C}$. Elements of the code are called *codewords*. The parameter d represents the minimum distance of the code, which is defined as the smallest Hamming distance between two codewords in the code, where the Hamming distance is Hamming weight of the difference between two codewords.

Definition: [*Syndrome Decoding Problem*] *Let $\mathcal{C}$ be some binary linear code whose parity check matrix is $H \in \mathbb{F}_n^{k\times n}$, and let $\mathbf{v} \in \mathbb{F}_2^k$ be some given vector. If $H\mathbf{y} = \mathbf{v}$, for some $\mathbf{y} \in \mathbb{F}_2^n$, then the vector $\mathbf{v}$ is said to be the syndrome of $\mathbf{y}$. We define the syndrome decoding instance $(H, \omega, \mathbf{v})$ as a problem of finding a vector $\mathbf{y} \in \mathbb{F}_2^n$ with weight ω such that $H\mathbf{y} = \mathbf{v}$.*

B. The Main Assumption

In this section we introduce and discuss our main assumption which is based on the hardness of syndrome decoding problem.

First let us start with some definitions.

1) Algorithm $Gen1$: Let $Gen1(n, k, l)$ be randomized polynomial-time algorithm which on input n, k and l returns the matrices $\tilde{H} \in \mathbb{F}^{k\times n}$, $\tilde{Y} \in \mathbb{F}^{k\times k}$ and $K \in \mathbb{F}^{n\times n}$, where

$$\tilde{Y} = R(I_k \oplus M^{\alpha_0} \oplus R_0 Q),\ \tilde{H} = \tilde{Y} S P^{-1},$$

$$S = \begin{bmatrix} M^{\alpha_1} & \cdots & M^{\alpha_\gamma} \end{bmatrix},$$

and

$$K = \begin{bmatrix} M^{-\alpha_1} & & \\ & \ddots & \\ & & M^{-\alpha_\gamma} \end{bmatrix},$$

for uniformly random full-rank matrices $R, R_0 \in \mathbb{F}_2^{k\times k}$, uniformly random permutation matrix $P \in \mathbb{F}_2^{n\times n}$, random matrix M as defined by Equation (1) with $Q = M^r \oplus I_k$, and uniformly random odd numbers $\alpha_j \in [1, \phi]$, $0 \le j \le \gamma$.

Theorem 2: Let $\tilde{Y} \leftarrow Gen1(n, k, l)$. Then, rank $\tilde{Y} = k - l'$.

Proof: For every $\mathbf{e}_i \in \mathcal{N}$, we have

$$\tilde{Y}\mathbf{e}_i = R(I_n \oplus M^{\alpha_0} \oplus R_0 Q)\mathbf{e}_i.$$

By Equation (3) we have $M^{\alpha_0}\mathbf{e}_i = \mathbf{e}_i$ and $Q\mathbf{e}_i = \mathbf{0}$. Hence,

$$\tilde{Y}\mathbf{e}_i = \mathbf{0}, \tag{4}$$

and $\mathbf{e}_i \in \text{Nul } \tilde{Y}$. Since $\mathbf{e}_i \in \mathcal{N} \Longrightarrow \mathbf{e}_i \in \text{Nul } \tilde{Y}$, it follows that $\mathcal{N} \subseteq \text{Nul } \tilde{Y}$.
Next, we know that $\mathcal{N}$ is l'-dimensional subspace. Therefore, $\mathcal{N} \subseteq \text{Nul } \tilde{Y}$ implies that

$$|\text{Nul } \tilde{Y}| \ge |\mathcal{N}| = 2^{l'}. \tag{5}$$

In other words, size of Nul $\tilde{Y}$ is not less than size of its subset $\mathcal{N}$.
On the other hand, we have by definition of the Nul of a matrix

$$\begin{aligned} |\text{Nul } \tilde{Y}| &= |\{\mathbf{x} : \tilde{Y}\mathbf{x} = \mathbf{0}\}| \\ &= |\{\mathbf{x} : R(I_n \oplus M^{\alpha_0})\mathbf{x} = RR_0Q\mathbf{x}\}| \\ &= |\{\mathbf{x} : R_0^{-1}(I_n \oplus M^{\alpha_0})\mathbf{x} = Q\mathbf{x}\}|, \end{aligned}$$

since each of R and R_0 is full-rank matrix and hence invertible. Since columns of the matrix Q live in the l'-dimensional space $\mathcal{N}$, there are at most $2^{l'}$ possible $\mathbf{x}$ such that $R_0^{-1}(I_n \oplus M^{\alpha_0})\mathbf{x} = Q\mathbf{x}$. Therefore,

$$|\text{Nul } \tilde{Y}| \le 2^{l'}. \tag{6}$$

From Equations (5) and (6) we see that $|\text{Nul } \tilde{Y}| = 2^{l'}$. Therefore,

$$\dim \text{Nul } \tilde{Y} = l' \text{ and rank } \tilde{Y} = k - l'.$$

■

Theorem 3: Let $(\tilde{H}, \tilde{Y}) \leftarrow Gen1(n, k, l)$. Then the matrix $\tilde{Y}$ is indistinguishable from the random. Furthermore, it is hard to find the matrix K or the permutation matrix P from $\tilde{H}$ and $\tilde{Y}$, for sufficiently large n.

Proof: Consider the matrix

$$\tilde{Y} = R(I_k \oplus M^{\alpha_0} \oplus R_0 Q).$$

It is obvious that the matrix $\tilde{Y}$ is product of two secret random matrices, where the left matrix R is uniformly random and the right matrix is

$$(I_k \oplus M^{\alpha_0} \oplus R_0 Q)$$

where R_0 secret uniformly random and also each of M^{α_0} and Q are secret and random. It is easy to see that columns of the matrix $\tilde{Y}$ are random linear combinations of columns of R which are in their turn uniformly random vectors in $\mathbb{F}_2^k$. Therefore, the matrix $\tilde{Y}$ is indistinguishable the random.
To prove the second part, consider

$$\tilde{H} = \tilde{Y} S P^{-1}$$

Suppose, for the sake of argument, that $\tilde{Y}$ is full-rank, thus one can easily recover the matrix

$$SP^{-1} = \left[\begin{array}{ccc} M^{\alpha_1} & \cdots & M^{\alpha_\gamma} \end{array} \right] P^{-1},$$

by solving the system of equations $\tilde{Y} S P^{-1} = \tilde{H}$ for the secret matrix SP^{-1}. However, both of S and P^{-1} are unknowns. Moreover, $\tilde{Y}$ is actually not full-rank matrix; as we see from Theorem 2, rank $\tilde{Y} = k - l'$.
This prevents full recovery of the matrix SP^{-1}, since any attempt of solving the system of equations $\tilde{Y} S P^{-1} = \tilde{H}$ for SP^{-1} will result in many solution. It is not hard to see that, since rank $\tilde{Y} = k - l'$, there are $2^{\Theta(n)}$ possible solutions for the system $\tilde{Y} S P^{-1} = \tilde{H}$.
Therefore, for sufficiently large n, it is hard to find any of the matrices K or P given only the matrices $\tilde{H}$ and $\tilde{Y}$. ∎
Next, let $\mathbf{h}_i = [\ \mathbf{h}_{i1} \quad \cdots \quad \mathbf{h}_{i\gamma}\ ]^T$, $\mathbf{h}_{ij} \in \mathbb{F}_2^k$, for $1 \le j \le \gamma$ and $i \in \mathbb{N}$. Thus $\bar{\mathbf{h}}_i = \sum_{j=1}^{\gamma} \mathbf{h}_{ij}$. Consider

$$\mathbf{y}_i = P(K\mathbf{h}_i \oplus \mathbf{r}_i), \text{ where } \mathbf{r}_i \in \mathcal{N}^\gamma.$$

That is, $\mathbf{r}_i = [\ \mathbf{r}_{i1} \quad \cdots \mathbf{r}_{i\gamma}\ ]^T$ and $\mathbf{r}_{ij} \in \mathcal{N}$, for $1 \le j \le \gamma$. Multiplying $\tilde{H}$ by $\mathbf{y}_i$ we get

$$\begin{aligned}
\tilde{H}\mathbf{y}_i &= \tilde{Y} \left[\begin{array}{ccc} M^{\alpha_1} & \cdots & M^{\alpha_\gamma} \end{array} \right] P^{-1} P(K\mathbf{h}_i \oplus \mathbf{r}_i) \\
&= \tilde{Y} \left[\begin{array}{ccc} M^{\alpha_1} & \cdots & M^{\alpha_\gamma} \end{array} \right] \times \\
&\left(\left[\begin{array}{ccc} M^{-\alpha_1} & & \\ & \ddots & \\ & & M^{-\alpha_\gamma} \end{array} \right] \left[\begin{array}{c} \mathbf{h}_{i1} \\ \vdots \\ \mathbf{h}_{i\gamma} \end{array} \right] \oplus \left[\begin{array}{c} \mathbf{r}_{i1} \\ \vdots \\ \mathbf{r}_{i\gamma} \end{array} \right] \right) \\
&= \tilde{Y} \left[\begin{array}{ccc} M^{\alpha_1} & \cdots & M^{\alpha_\gamma} \end{array} \right] \left[\begin{array}{c} M^{-\alpha_1}\mathbf{h}_{i1} \oplus \mathbf{r}_{i1} \\ \vdots \\ M^{-\alpha_\gamma}\mathbf{h}_{i\gamma} \oplus \mathbf{r}_{i\gamma} \end{array} \right] \\
&= \tilde{Y} \sum_{j=1}^{\gamma} M^{\alpha_j}(M^{-\alpha_j}\mathbf{h}_{ij} \oplus \mathbf{r}_{ij}) \\
&= \tilde{Y} \sum_{j=1}^{\gamma} (\mathbf{h}_{ij} \oplus M^{\alpha_j}\mathbf{r}_{ij}), \\
&\text{(by Equation (3), we have } M^{\alpha_j}\mathbf{r}_{ij} = \mathbf{r}_{ij}) \\
&= \tilde{Y} \sum_{j=1}^{\gamma} \mathbf{h}_{ij} \oplus \sum_{j=1}^{\gamma} \tilde{Y}\mathbf{r}_{ij} \\
&= \tilde{Y}\bar{\mathbf{h}}_{\mathbf{i}},
\end{aligned}$$

where $\bar{\mathbf{h}}_i = \sum_{j=1}^{\gamma} \mathbf{h}_{ij}$ and $\tilde{Y}\mathbf{r}_{ij} = \mathbf{0}$ for each j, $1 \le j \le \gamma$ (by Equation (4)).
Therefore,

$$\tilde{H}\mathbf{y}_i = \tilde{Y}\bar{\mathbf{h}}_i. \tag{7}$$

2) Algorithm $Gen2$: Let $Gen2(n,k,l)$ be randomized polynomial-time algorithm which on input (n,k,l) returns the matrices $Y \in \mathbb{F}_2^{(k-l')\times k}$, $H = YSP^{-1}$, K and P, where Y is a matrix that consists of $k-l'$ linearly independent rows of $\tilde{Y}$, and $(\tilde{Y}, S, K, P) \leftarrow Gen1(n,k,l)$.

Recall that rank $\tilde{Y}$ is $k-l'$. That is, $\tilde{Y}$ has $(k-l')$ linearly independent rows. Let $k' = k - l'$. Thus, $Y \in \mathbb{F}_2^{k'\times k}$ and $H \in \mathbb{F}_2^{k'\times n}$. Since rows of Y are subset of rows of $\tilde{Y}$ we have

$$H\mathbf{y}_i = Y\bar{\mathbf{h}}_i. \tag{8}$$

Now we introduce our main assumption.

Assumption 1: Given the matrices H and Y and the sequence of pairs

$$\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i} = \{(\mathbf{h}_i, \mathbf{y}_i) \in \mathbb{F}_2^n \times \mathbb{F}_2^n : \mathbf{y}_i = P(K\mathbf{h}_i \oplus \mathbf{r}_i)\}_{i=1}^{q},$$

for uniformly random vectors $\mathbf{h}_1, \ldots, \mathbf{h}_q$ from $\mathbb{F}_2^n$, random independent vectors $\mathbf{r}_1, \ldots, \mathbf{r}_q$ from $\mathcal{N}^\gamma$, such that the Hamming weight of each $\mathbf{y}_i$ is $wt(\mathbf{y}_i) \approx \omega$. Then, given random $\mathbf{h} \in \mathbb{F}_2^n$, it is hard to find a vector $\mathbf{y} \in \mathbb{F}_2^n$ with weight $wt(\mathbf{y}) \approx \omega$ such that

$$H\mathbf{y} = Y\bar{\mathbf{h}}. \tag{9}$$

Let $\mathcal{A}$ be probabilistic polynomial-time adversary, we state Assumption 1 more neatly in terms of advantage function of the adversary $\mathcal{A}$ in finding a pair $(\mathbf{h}, \mathbf{y}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ which satisfies Equation (9) as

$$Adv(\mathcal{A}) = Pr[H\mathbf{y} = Y\bar{\mathbf{h}} : (\mathbf{h}, \mathbf{y}) \leftarrow \mathcal{A}(H, Y, \mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i})],$$

where $\mathbf{h}$ is generated by some random oracle.

Under hardness of the syndrome decoding instance $(H, \omega, \mathbf{v} = Y\bar{\mathbf{h}})$, we know that, for a random vector $\mathbf{h} \in \mathbb{F}_2^n$, it is hard for the adversary to find a vector $\mathbf{y} \in \mathbb{F}_2^n$ (with weight $\approx \omega$) that satisfies $H\mathbf{y} = Y\bar{\mathbf{h}}$. Because doing so would imply that the adversary can solve the syndrome decoding instance $(H, \omega, \mathbf{v})$ which is hard problem.

C. Justification of Hardness of the Assumption 1

1) Secrecy of the Matrices K and P: Given the matrices H and Y from Algorithm $Gen2$ and the sequence $\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i} =$

$$(\mathbf{h}_1, \mathbf{y}_1 = P(K\mathbf{h}_1 \oplus \mathbf{r}_1)), \ldots, (\mathbf{h}_i, \mathbf{y}_q = P(K\mathbf{h}_q \oplus \mathbf{r}_q)),$$

where $\mathbf{r}_1, \ldots, \mathbf{r}_q$ are uniformly random secret vectors from $\mathcal{N}^\gamma$. For a random vector $\mathbf{h} \in \mathbb{F}_2^n$, without having the secret matrices P and K, one cannot compute $\mathbf{y} = P(K\mathbf{h} \oplus \mathbf{r})$, for some $\mathbf{r} \in \mathcal{N}^\gamma$.
From Theorem 3 we see that Y is computationally indistinguishable from the random and also it is hard to find K and P given H and Y.
We furthermore conjecture that, given H, Y and the sequence

$\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i}$, it is hard to recover the secret matrices P and K. The argument proceeds as follows.

To begin with, note that unlike permuting a vector or a list of n distinct elements, permuting coordinates a binary vector whose entries take only one of two possible values (i.e., 0 or 1) is hard to observe.

2) Example: To give an illustrative example, let $\mathbf{x} = (1, 2, 3, 4)$ and $\mathbf{y} = (3, 2, 4, 1)$, if $\mathbf{y} = P\mathbf{x}$ then it easy to figure out the permutation P. However, if for example $\mathbf{x} = (1, 0, 0, 1)$, $\mathbf{y} = (0, 1, 0, 1)$, and $\mathbf{y} = P\mathbf{x}$, then it is not easy to tell what is the permutation P, because there is more than one permutation.

That is, in the case of the binary vector, given for example $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$ such that $wt(\mathbf{x}) = wt(\mathbf{y})$, there are many possible permutations P such that $\mathbf{y} = P\mathbf{x}$. This generally makes permutation of binary vectors harder to detect than permutation of a list with n distinct element. (Note that in our assumption both $\mathbf{x}$ and P are unknowns.)

Now we show that the sequence $\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i} =$

$$(\mathbf{h}_1, \mathbf{y}_1 = P(K\mathbf{h}_1 \oplus \mathbf{r}_1)), \ldots, (\mathbf{h}_i, \mathbf{y}_q = P(K\mathbf{h}_q \oplus \mathbf{r}_q))$$

reveals no information about the matrices K and P. To see this note that each secret vector $\mathbf{r}_i$ is selected uniformly at random. Hence, the vector $(K\mathbf{h}_i \oplus \mathbf{r}_i)$ is uniformly random and secret for each i (note that the set $\mathcal{N}^\gamma$ from which we draw $\mathbf{r}_i$ has size $|\mathcal{N}^\gamma| = 2^{l'\gamma}$), and therefore the vector $\mathbf{y}_i = P(K\mathbf{h}_i \oplus \mathbf{r}_i)$ which is matrix-vector product of secret permutation matrix P and secret vector $(K\mathbf{h}_i \oplus \mathbf{r}_i)$, does not reveal any information about the secret matrix K or the secret permutation matrix P. Furthermore, for $i \neq j$, each of $\mathbf{h}_i$ and $\mathbf{r}_i$ is, by definition, independent form $\mathbf{h}_j$ and $\mathbf{r}_j$, respectively. Hence, $K\mathbf{h}_i \oplus \mathbf{r}_i$ is independent from $K\mathbf{h}_j \oplus \mathbf{r}_j$. That is, the vectors $K\mathbf{h}_i \oplus \mathbf{r}_i$, for $1 \leq i \leq q$, are secret random independent vectors. Therefore, the sequence $\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i}$ reveals no information about K or P.

3) Hardness of Syndrome Decoding Problem: Once again the elements $(\mathbf{h}_i, \mathbf{y}_i)$ of the sequence $\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i}$ are independent from each other. To see this note that, by their definition, the vectors $\mathbf{h}_1, \ldots, \mathbf{h}_q$ are random independent vectors. Also, by definition, the vectors $\mathbf{r}_1, \ldots, \mathbf{r}_q$ are random independent vectors. Hence, each $\mathbf{y}_i = P(K\mathbf{h}_i \oplus \mathbf{r}_i)$ is independent from $\mathbf{y}_j = P(K\mathbf{h}_j \oplus \mathbf{r}_j)$, for $\mathbf{h}_i \neq \mathbf{h}_j$, simply because each of $\mathbf{h}_i$ and $\mathbf{r}_i$ are independent from $\mathbf{h}_j$ and $\mathbf{r}_j$, respectively. Therefore, given a particular random vector $\mathbf{h} \in \mathbb{F}_2^n$,

$$Pr[H\mathbf{y} = Y\bar{\mathbf{h}} : (\mathbf{h}, \mathbf{y}) \leftarrow \mathcal{A}(H, Y, \mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i})]$$

$$=$$

$$Pr[H\mathbf{y} = Y\bar{\mathbf{h}} : (\mathbf{h}, \mathbf{y}) \leftarrow \mathcal{A}(H, Y)].$$

In other words, providing the adversary with the sequence $\mathcal{S}_{\mathbf{h}_i,\mathbf{y}_i}$ does not improve his advantage in finding a vector $\mathbf{y}$ such that $H\mathbf{y} = Y\bar{\mathbf{h}}$.

As mentioned before, for any given $\mathbf{h} \in \mathbb{F}_2^n$, if the syndrome decoding instance $(H, \omega, Y\bar{\mathbf{h}})$ is hard, then it is hard to find a vector $\mathbf{y} \in \mathbb{F}_2^n$ which has weight ω such that $H\mathbf{y} = Y\bar{\mathbf{h}}$. The matrix $H \in \mathbb{F}_2^{k' \times n}$ represents subset of rows of parity check matrix for some (unknown) random linear $[n, k', d]$-code $\mathcal{C}$ with distance $d \geq \omega$, and the vector $Y\bar{\mathbf{h}}$ represents a syndrome for some unknown erroneous codeword $\mathbf{c}' \in \mathcal{C}$. And we see by Theorem 3 that Y is computationally indistinguishable from the random. Also, $H = YSP^{-1}$ for secret random matrices S and P. Thus, columns of the matrix H are just random linear combinations of columns of the random matrix Y. Therefore, finding a vector $\mathbf{y}$ such that $H\mathbf{y} = Y\bar{\mathbf{h}}$ means solving the syndrome decoding instance $(H, \omega, Y\bar{\mathbf{h}})$, which is well-known hard problem for sufficiently large n and appropriately restricted ω.

4) Upshot: Combining (a) secrecy of the matrices K and P, (b) uniformity and independence of elements of the sequence $(\mathbf{h_i}, \mathbf{y}_i)$, for $i \in [1, q]$, and (c) hardness of the syndrome decoding instance $(H, \omega, Y\bar{\mathbf{h}})$, we conclude that Assumption 1 is hard.

V. Digital Signature Scheme

In this section we introduce our digital signature algorithm which is based on the Assumptions 1.

1) Primitive Components of the Scheme: The scheme consists of four basic components. The most basic building block of the scheme is a structured nonsingular matrix $M \in \mathbb{F}_2^{k \times k}$. The second component is a random subspace of $\mathbb{F}_2^n$, which is referred to as *noise distribution associated with the matrix* M, denoted $\mathcal{N}^\gamma$, and from which we draw our random noise vectors, hence we call it the *Noise Distribution*. The third component is a function $\mathcal{R}(\cdot)$ which is used to reduce the Hamming weight of the signature to the required bound; thus it takes as its input a vector $\mathbf{u} \in \mathbb{F}_2^n$ and returns reduced weight vector $\mathbf{u}' = \mathcal{R}(\mathbf{u}, \tilde{B}, t)$. The function $R(\cdot)$ reduces weight of the vector $\mathbf{u} \in \mathbb{F}_2^n$ by adding to it some vectors from the noise distribution $\mathcal{N}^\gamma$ such that the resulting vector is ensured to have approximate weight ω.

2) Definition: [*Digital Signature*] *A Digital Signature Scheme is a scheme specified by three polynomial-time algorithms* $KegGen(1^\lambda)$ *,*$Sign(pk, m)$ *and,* $Verify(sk, (m, s))$*, 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.*
- $Sign(sk, m)$ *is a randomized algorithm that takes the private key* sk *and a message* m *as an input and returns the pair* (m, s)*, of the message* m *and its corresponding signature* s*.*
- $Verify(pk, (m, s))$ *is a deterministic algorithm that takes the secret key* pk *the pair* (m, s') *and response by "Accept" if* s' *is valid signature for the message* m*, or "Reject" otherwise.*

In what follows we present the proposed digital signature scheme in details.

3) Algorithm $KeyGen$*:* We start with $KeyGen$ which is a probabilistic polynomial-time algorithm that on input 1^n returns a pair of (*public*, *private*) keys

$$(pk, sk) = ((H, Y, \mathcal{H}, d, \omega, \mu), (K, P)),$$

where $(H, Y, K, P) \leftarrow Gen2(n, k, l)$ exactly as in the Assumption 1, $\mathcal{H}$ is a hash function such that $\mathcal{H} : \{0,1\}^* \longrightarrow \mathbb{F}_2^n$, d is distance of $[n, k', d]$ binary linear code, ω is Hamming weight of the valid signature, and μ is some small positive integer which will be used as an approximation margin for ω.

Algorithm 2 $KeyGen(1^n)$

Require: n.
Ensure: $(pk, sk) = ((H, Y, \mathcal{H}, d, \omega, \mu), (K, P))$.
1: Set $\gamma \leftarrow 4$, $\delta \leftarrow 0.9$;
2: Set $k \leftarrow \frac{n}{\gamma}$, $l \leftarrow \delta k$, and $k' \leftarrow \frac{n}{\gamma}(1 - \frac{\delta}{2})$;
3: Generate $(H, Y, K, P) \leftarrow Gen2(n, k, l)$;
4: Specify hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \mathbb{F}_2^n$;
5: Set $d \leftarrow n\mathsf{H}^{-1}(1 - \frac{k'}{n})$, $\omega \leftarrow \frac{n}{2}(1 - \frac{\delta}{2})$, and $\mu \leftarrow \lceil \log n \rceil$;
▷ H^{-1} is inverse of the binary entropy function.
6: Output $(pk, sk) = ((H, Y, \mathcal{H}, d, \omega, \mu), (K, P))$;

Observe that, by definition of Algorithm $Gen2$, H is $k' \times n$ matrix with k' linearly independent rows.

4) Size of the Private Key (The Standard Mode): Recall that secret matrix K is direct sum of four matrices $M^{-\alpha_1}$, $M^{-\alpha_2}$, $M^{-\alpha_3}$, and $M^{-\alpha_4}$. That is

$$K = \begin{bmatrix} M^{-\alpha_1} & & & \\ & M^{-\alpha_2} & & \\ & & M^{-\alpha_3} & \\ & & & M^{-\alpha_4} \end{bmatrix}$$

In order to minimize memory consumption we select uniformly random odd number $\alpha_1 \in [1, \phi - 9]$, then (without loss of generality) we set $\alpha_i \leftarrow 2 + \alpha_{i-1}$ for $2 \leq i \leq 4$. Instead of storing the matrix K which $n \times n$ we only store the matrix M^{-1} which is $k \times k$ and the secret exponent α_1. Also, the secret permutation matrix P is stored as an $n-$tuple permutation (or array). For ranges of n in the scheme this $n-$tuple permutation is stored in $2n$ bytes. Furthermore, we do not need to reconstruct the matrix K at all. We carry out the matrix-vector multiplication $K\mathbf{h}$, where $\mathbf{h} = [\ \mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3 \quad \mathbf{h}_4\ ]^T$ and $\mathbf{h}_i \in \mathbb{F}_2^k$ in signing algorithm using the sub-matrices $M^{-\alpha_i}$, since $M^{-\alpha_i}\overline{\mathbf{h}}_i = M^{-(2+\alpha_{i-1})}\overline{\mathbf{h}}_i$. Thus, size of the private key is sum of: k^2 bits (size of M^{-1}) plus $2n$ bytes (size of the permutation P) plus $(m = (k - l))$ bits (size of α_1). Note that $k - l = k - \delta k = 0.10k$. That is, size of sk is

$$(k^2 + 16n + 0.10k) \text{ bits.}$$

This optimization in memory usage comes at cots of run-time since it requires more matrix-vector multiplications. But this cost is acceptable given the reduction in size of the private key.

Signing and verifying are performed as described in the following two algorithms.

5) Algorithm $Sign$: To sign a message m, we first compute the hash value $\mathbf{h} \leftarrow \mathcal{H}(m)$ and then compute $\mathbf{u} \leftarrow K\mathbf{h}$. Now, the vector $\mathbf{u} \in \mathbb{F}_2^n$ is secret random vector with Hamming weight approximately $\frac{n}{2}$. Next, we apply the function $\mathcal{R}(\cdot)$ on $\mathbf{u}$ to obtain $\mathbf{u}' \leftarrow \mathcal{R}(\mathbf{u}, \tilde{B}, t)$ which has weight $wt(\mathbf{u}') \approx \omega$. Also, (see Remark III-C2) the vector $\mathbf{u}'$ has the form $\mathbf{u}' = K\mathbf{h} \oplus \mathbf{r}$, for some unknown random vector $\mathbf{r} \in \mathcal{N}^\gamma$. Finally, we compute the signature $\mathbf{s} \leftarrow P\mathbf{u}'$.
We know that the weight of the signature is $wt(\mathbf{s}) \approx \omega$, since the multiplication by permutation matrix P preserves the weight. Since $\mathbf{u}'$ is secret random vector, the signature $\mathbf{s} = P\mathbf{u}'$ does not reveal the secret permutation matrix P, and also the matrix K remains secret, no mater how many number of signatures; every time the signature is $\mathbf{s} = P\mathbf{u}'$ for a secret random vector $\mathbf{u}' \in \mathbb{F}_2^n$ and secret permutation matrix P.

Algorithm 3 $Sign$

Require: (m, sk).
Ensure: $(m, \mathbf{s})$.
1: $\mathbf{h} \leftarrow \mathcal{H}(m)$;
2: $\mathbf{u} \leftarrow K\mathbf{h}$;
3: $t \leftarrow l'$;
4: $\mathbf{u}' \leftarrow \mathcal{R}(\mathbf{u}, \tilde{B}, t)$;
5:
6: **while** $wt(\mathbf{u}') < \omega - \mu$ **or** $wt(\mathbf{u}') < d - \mu$ **do**:
7: $t \leftarrow (t - 1)$;
8: $\mathbf{u}' \leftarrow \mathcal{R}(\mathbf{u}, \tilde{B}, t)$;
9: **end while**
10: $\mathbf{s} \leftarrow P\mathbf{u}'$;
11:
12: **if** $wt(\mathbf{s}) < \omega - \mu$ **or** $wt(\mathbf{s}) > d + \mu$ **then**
13: return $\perp$ and exit; ▷ Report failure.
14: **end if**
15: Output $(m, \mathbf{s})$;

Since the hash value $\mathbf{h} \in \mathbb{F}_2^n$ is uniformly random and K is full-rank $n \times n$ matrix, the secret vector $\mathbf{u} = K\mathbf{h}$ is uniformly random with approximate weight $\frac{n}{2}$. Therefore, by definition of the function $\mathcal{R}(\cdot, \tilde{B}, t)$ and Theorem 1, the vector $\mathbf{u}' = \mathcal{R}(\mathbf{u}, \tilde{B}, t = l')$ in step (4) of the Algorithm $Sign$ will have Hamming weight $wt(\mathbf{u}') \approx \frac{\gamma(k-l')}{2} = \frac{n}{2}(1 - \frac{\delta}{2}) = 0.275n$.

6) Remark: Note that

$$\omega = \frac{n}{2}(1 - \frac{\delta}{2}) \text{ and } wt(\mathbf{u}') \approx \frac{n}{2}(1 - \frac{\delta}{2}).$$

That is $wt(\mathbf{u}') \approx \omega$. Since $\omega < d$, the vector $\mathbf{u}'$ will satisfy $\omega - \mu \leq wt(\mathbf{u}') \leq d + \mu$ with overwhelming probability. However, if necessary the parameter μ can be increased for more certainty, but it does not need to be large to ensure that $\omega - \mu \leq wt(\mathbf{u}') \leq d + \mu$.

The vector $\mathbf{u}$ initially has weight $wt(\mathbf{u}) \approx \frac{n}{2}$. Thus, from the beginning, there is no problem with the lower bound of $wt(\mathbf{u})$ and hence we do not to worry about $wt(\mathbf{u}')$ being less than $\omega - \mu$. We always need to reduce weight of $\mathbf{u}$ in order to ensure that $wt(\mathbf{s}) \leq d + \mu$. We use the function $\mathcal{R}(\cdot, \tilde{B}, t)$ which can reduce the weight up to $0.275n$ (when $t = l'$).
However, if $wt(\mathbf{s}) \leq \omega - \mu$, then we repeatedly set $t \leftarrow (t-1)$ and recompute $\mathbf{u}' \leftarrow \mathcal{R}(\mathbf{u}, \tilde{B}, t)$ until $\omega - \mu \leq wt(\mathbf{s})$. Note that in each iteration weight of the vector $\mathbf{u}'$ is $wt(\mathbf{u}') = \frac{\gamma(k-t)}{2}$ (by Theorem 1). Since t is decreasing the weight $wt(\mathbf{u}') = \frac{\gamma(k-t)}{2}$ will eventually exceed $\omega - \mu$ and the **while** loop ends.

Next, the matrix P in step (10) is permutation matrix and hence the matrix-vector multiplication $P\mathbf{u}'$ does not change the weight of the vector $\mathbf{u}'$. Therefore, weight of the signature vector $\mathbf{s} = P\mathbf{u}'$ is $wt(\mathbf{s}) = wt(\mathbf{u}')$. Hence

$$\omega - \mu \leq wt(\mathbf{s}) \leq d + \mu.$$

However, if otherwise $wt(\mathbf{s}) < \omega - \mu$ or $wt(\mathbf{s}) > d + \mu$, then the algorithm reports failure, which is negligibly small probability since $wt(\mathbf{u}') \approx \omega$ and $\omega < d$. (See Section 5 for concrete values of d, ω and μ.)
Finally, recall form Remark III-C2 that the vector $\mathbf{u}'$ has the form $\mathbf{u}' = \mathbf{u} \oplus \mathbf{r}$, where $\mathbf{r} \in \mathcal{N}^{\gamma}$. Since $\mathbf{u} = K\mathbf{h}$, we have

$$\mathbf{s} = P\mathbf{u}' = P(\mathbf{u} \oplus \mathbf{r}) = P(K\mathbf{h} \oplus \mathbf{r}).$$

Thus, the signature $\mathbf{s}$ satisfies conditions of the Assumption 1; it has restricted weight $wt(\mathbf{s}) \approx \omega$, and it also has structure $\mathbf{s} = P(K\mathbf{h} \oplus \mathbf{r})$, where $\mathbf{r} \in \mathcal{N}^{\gamma}$.

7) Algorithm $Verf$: A valid signature $\mathbf{s}$ must satisfy

$$H\mathbf{s} = Y\bar{\mathbf{h}} \text{ and } wt(\mathbf{s}) \approx \omega.$$

Algorithm 4 Verf

Require: $(m, \mathbf{s}, pk)$.
Ensure: Accept or reject the the signature $\mathbf{s}$.
1: $\mathbf{h} \leftarrow \mathcal{H}(m)$;
2:
3: **if** $\omega - \mu \leq wt(\mathbf{s}) \leq d + \mu$ and $H\mathbf{s} = Y\bar{\mathbf{h}}$ **then** output 1,
4: **else** output 0;
5: **end if**

Thus, for a particular hash value $\mathbf{h}$, forging a signature $\mathbf{s}' \in \mathbb{F}_2^n$ is equivalent to finding a vector $\mathbf{s}'$ that solves the syndrome decoding instance

$$(H, \omega, Y\bar{\mathbf{h}}).$$

8) Correctness of the Signature Verification: Correctness of the valid signature follows from Equation (8) and Remark V-6.

VI. Security of the Scheme

In this section we show that, if the Assumption 1 holds, then the proposed digital signature scheme is secure against existential forgery under adaptive chosen plaintext attack.
Indeed, Assumption 1 can be reformulated into definition of *security against existential forgery under adaptive chosen plaintext attack* for our signature scheme. In this definition [11], the adversary is given an oracle access to the signing algorithm, so that he can interact adaptively with algorithm $Sign(\cdot, sk)$ issuing polynomially bounded number q of signing queries, and eventually coming up with new message m that has not been signed before, trying to forge a valid signature for it.

1) Adversarial Model of Assumption 1: Let $\mathcal{A}$ be probabilistic polynomial-time adversary who has the public matrices (H, Y). We assume that the adversary has access to two oracles; a random oracle $\mathcal{O}_{\mathbf{h}}$ which generates uniformly random vectors $\mathbf{h}_i \in \mathbb{F}^n$, and signing oracle $Sign(\cdot, sk)$ which takes the vector $\mathbf{h}_i$ as an input and returns the vector $\mathbf{y}_i = P(K\mathbf{h}_i \oplus \mathbf{r}_i)$, which represents the signature, where $\mathbf{r}_i \xleftarrow{R} \mathcal{N}^{\gamma}$. The adversary's goal is to find a new pair $(\mathbf{h}, \mathbf{s})$ in which $\mathbf{h}$ has not been sent to the oracle $Sign(\cdot, sk)$ before such that $wt(\mathbf{s}) \approx \omega$ and

$$H\mathbf{s} = Y\bar{\mathbf{h}}.$$

We define advantage function of $\mathcal{A}$ in achieving his goal as follows

$$Pr\left[H\mathbf{s} = Y\bar{\mathbf{h}} : \begin{array}{c} Adv_{frg}(\mathcal{A}) = \\ (H, Y, K, P) \leftarrow Gen2(n, k, l) \\ (\mathbf{h}, \mathbf{s}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathbf{h}}, Sign(\cdot, sk)}(H, Y) \end{array}\right].$$

Definition: Let $\mathcal{A}$ be probabilistic polynomial-time adversary who has an oracle access to the algorithm $Sign$. We define an advantage function of $\mathcal{A}$ in breaking our signature scheme as follows.

$$Pr\left[H\mathbf{s} = Y\bar{\mathbf{h}} : \begin{array}{c} Adv^{acma}_{frg}(\mathcal{A}) = \\ (sk, pk) \leftarrow KeyGen(1^n); \\ (m, \mathbf{s}) \leftarrow \mathcal{A}^{Sign(\cdot, sk)}(pk); \end{array}\right].$$

Note that in spite of the fact the adversary is issuing signing queries for a sequence of messages $m_1, \ldots, m_q$ adaptively, yet the corresponding hash values $\mathbf{h}_1, \ldots, \mathbf{h}_q$ are produced randomly (out of the adversary's control). The hash function $\mathcal{H}(\cdot)$ whose output for every message m is out of control of the adversary, is modeled in the adversarial model by the random oracle $\mathcal{O}_{\mathbf{h}}$ which returns uniformly random vector $\mathbf{h} \in \mathbb{F}_2^n$. Hence, the signing queries from $\mathcal{A}$ and the corresponding signatures generated by the signing oracle $Sign(\cdot, sk)$ are captured by the sequence $(\mathbf{h}_1, \mathbf{y}_q), \ldots, (\mathbf{h}_q, \mathbf{y}_q)$ in the Assumptions 1 where $\mathbf{h}_1, \ldots, \mathbf{h}_q$ is uniformly random sequence of vectors in $\mathbb{F}_2^n$. That is, the adaptive interaction of the adversary with algorithm $Sign$ is captured by the sequence

$$\mathcal{S}_{\mathbf{h}_i, \mathbf{y}_i} = (\mathbf{h}_1, \mathbf{y}_q), \ldots, (\mathbf{h}_q, \mathbf{y}_q)$$

in Assumption 1. Hence, breaking the signature scheme directly implies breaking Assumption 1.
In other words, $Adv^{acma}_{frg}(\mathcal{A}) \equiv Adv_{frg}(\mathcal{A})$. Therefore, Assumption 1 directly implies that $Adv^{acma}_{frg}(\mathcal{A}) \approx neg(n)$ for sufficiently large n.

Theorem 4: Under Assumption 1, the signature scheme $(KeyGen, Sign, Verf)$ is secure against existential forgery under adaptive chosen plaintext attack.

Proof: The proof follows directly from the equivalence between $Adv^{acma}_{frg}(\mathcal{A})$ and $Adv_{frg}(\mathcal{A})$. ■

A. Security Parameters for Assumption 1

Let $(H, Y) \leftarrow Gen2(n, k, l)$. Note that

$$Y \in \mathbb{F}_2^{k' \times k} \text{ and } H \in \mathbb{F}_2^{k' \times n}.$$

TABLE III
PARAMETER SETS FOR STANDARD SSD

Parameter set	SSD1760	SSD2744	SSD3728
γ	4	4	4
δ	0.9	0.9	0.9
$[n, k', d]$	$[1760, 242, 502]$	[2744,337,782]	$[3728, 512, 1063]$
ω	484	754	1025
μ	11	12	12
k	440	686	932

Suppose that $\mathcal{C}$ is random binary linear $[n, k', d]$-code. Let

$$d = n\mathsf{H}^{-1}(1 - \frac{k'}{n}),$$

where $\mathsf{H}^{-1}(\cdot)$ is inverse of the binary entropy function

$$\mathsf{H}(x) = -x\log(x) - (1-x)\log(1-x).$$

Consider the syndrome decoding instance

$$(H, \omega, \mathbf{v}), \text{ where } \omega \leq d. \tag{10}$$

We see that Assumption 1 depends on the hardness of solving an instance $(H, \omega, \mathbf{v})$ of the syndrome decoding problem which asks for vector $\mathbf{y} \in \mathbb{F}_2^n$ with weight ω such that $H\mathbf{y} = \mathbf{v}$.

The relation between decoding the random linear $[n, k', d]$-code $\mathcal{C}$ and solving the syndrome decoding instance $(H, \omega, \mathbf{v})$ where H is $k' \times n$ lies in the fact that the matrix H can be viewed as part of a larger (unknown) $(n-k') \times n$ parity check matrix that defines the code $\mathcal{C}$. That is, rows of our matrix H are viewed as subset of rows of a larger (unknown) $(n-k') \times n$ parity check matrix for an (unknown) code $\mathcal{C}$, and we view the vector $(\mathbf{v} = \bar{\mathbf{h}})$ as a syndrome for some (unknown) code word of $\mathcal{C}$ which is usually used to correct errors, but we are not concerned with an error correction problem in our scheme. We are not interested in the code $\mathcal{C}$; we only use its parameter (the block length n, the dimension dimension k', and the distance d) as indicators for hardness of the syndrome decoding instance $(H, \omega, \mathbf{v})$ where $\omega \approx d$.

1) Parameter Sets for Standard SSD*:* The syndrome decoding problem is well-known hard problem that has long history of study. Solving algorithms for this problem generally has exponential time complexity $2^{O(k')}$ for restricted weight $\omega \leq d$, where d is determined by the Gilbert-Varshamov [10] bound: $d = n\mathsf{H}^{-1}(1 - \frac{k'}{n})$.

Hardness of the syndrome decoding instance for a $[n, k', d]$-code linear code $\mathcal{C}$ increases proportionally with dimension of the code k' and weight of the syndrome ω. Therefore, we aim at maximizing both k' and ω while keeping $\omega \leq d$. So, we choose the parameters n, γ and δ such that $\omega \leq d$ and k' as large as possible.

Let $k = \frac{n}{\gamma}$ and $l = \delta k$. Hence $l' = \frac{l}{2} = \frac{\delta k}{2} = \frac{\delta n}{2\gamma}$ (see Equation (1)). We set $\omega \leftarrow \frac{\gamma(k-l')}{2}$. Thus,

$$\omega = \frac{\gamma(k-l')}{2} = \frac{n}{2}(1 - \frac{\delta}{2}). \tag{11}$$

TABLE IV
SIZES OF THE KEYS AND THE SIGNATURE FOR DIFFERENT SECURITY LEVELS OF THE STANDARD *SSD*.

Security Level	128-bits	192-bits	256-bits
Parameter Set	SSD1760	SSD2744	SSD3728
Public Key Size (KB)	65	158	291.5
Private Key Size (KB)	27	63	113
Signature Size (Bytes)	220	343	466

Since $k' = k - l'$, it follows that

$$k' = k(1 - \frac{\delta}{2}) = \frac{n}{\gamma}(1 - \frac{\delta}{2}). \tag{12}$$

Thus, $\frac{k'}{n} = \frac{1}{\gamma}(1 - \frac{\delta}{2})$.

We choose $\gamma = 4$ and $\delta = 0.9$ in order to achieve the optimal values for k' and ω. Thence we specify three parameter sets (Table III) SSD1760,SSD2744 and SSD3728 which are defined by the three basic parameters γ, δ, and n; the rest of the parameters are calculated as functions of these basic parameters.

VII. SYNDROME DECODING ESTIMATOR

In [7] Andre Esser and Emanuele Bellini give concrete hardness estimations for time complexity for the best known solving algorithms for syndrome decoding problem. That is, their work provides us with concrete estimation for hardness any given instance of 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 develop a framework that allows them to obtain several of the major information set decoding algorithms by choosing specific configurations of that framework. Then, these algorithms are analyzed to drive formulas that give the concrete complexity of solving the syndrome decoding problem for each algorithm. Furthermore, Esser and Bellini 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 for code $[n, k', d]$-code $\mathcal{C}$ with weight $\omega \approx d$, then we select our security parameters accordingly.

However, we should mention here that the matrix H in our scheme has less number of rows than a parity check matrix that defines an $[n, k', d]$-code (about 6.5 times less). So, the reader should take this in consideration. Also, following caution of Esser and Bellini we should mention that these estimations serve mainly as indicative benchmarks for ranges of parameter

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

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 [7].

Tables V, VI and VII show Esser-Bellini's time complexity and memory estimation of syndrome decoding of each parameter set (security level) of our scheme. Since we allow weight of the signature $wt(\mathbf{s})$ to be within the range $\omega-\mu \leq wt(\mathbf{s}) \leq d+\mu$, we calculated estimation of hardness of three instances for each level: $(n, k', \omega-\mu)$, (n, k', ω), and $(n, k', d+\mu)$ to make sure the whole range of possible values of $wt(\mathbf{s})$ satisfies the required security level. Also, from these estimations we see that the range $[\omega-\mu, d+\mu]$ can be expanded by increasing μ with no significant decreasing of the relevant security level, but for now we set $\mu \leftarrow \lceil \log n \rceil$.

VIII. Light Verification Key (LVK) Mode

In this section, we introduce *Light Verification Key Mode* of operation (LVK Mode) in order to improve performance of the verification in terms of both memory and run-time without affecting the security. At the verifying end, given the public key

$$pk = (H, Y, \mathcal{H}, d, \omega, \mu)$$

the verifying party secretly forms the matrices H' and Y' from rows of H and Y, respectively, by selecting random k'' rows from the matrix H and k'' of their corresponding counterparts from Y, where $k'' < k'$. Then the verifying party forms the new *light verification key*

$$pk^{'} = (H', Y', \mathcal{H}, d, \omega, \mu)$$

and keeps it privately. Now, instead of keeping the entire public key pk, the the verifying party keeps only the smaller size light key pk', where H' is $k'' \times n$ matrix and Y' is $k'' \times k$ matrix, instead of $k' \times n$ and $k' \times k$ in H and Y respectively in the original public key. The verification algorithm run as usual with H and Y replaced by H' and Y' respectively. If the signature $\mathbf{s}$ is valid, then correctness of the equation $H'\mathbf{s} = Y'\bar{\mathbf{h}}$ follows directly from the fact that rows of H' and Y' already satisfy the equation $H\mathbf{s} = Y\bar{\mathbf{h}}$. Thus, the verifying party accepts the signature $\mathbf{s}$ as a valid signature if

$$\omega - \mu \leq wt(\mathbf{s}) \leq d + \mu$$

and the signature $\mathbf{s}$ solves the syndrome decoding instance

$$(H', \omega, Y'\bar{\mathbf{h}}).$$

Since H' and Y' has fewer rows, the instance $(H', \omega, Y'\bar{\mathbf{h}})$ is easier to solve (depending on how small k'' is). However, since the verifying key is random and secret, the adversary does not know it, and therefore it is hard for him to forge a valid signature that solves that secret verification instance (H, k'', ω).

The adversary knows the original public key, but he does not know which part of it will be used by the verifying party. Since the matrices H' and Y' are constructed by choosing uniformly random rows of H and Y respectively, with

$$\binom{k'}{k''} = \frac{k'!}{k''!(k'-k'')!}$$

possible secret matrices H' and Y'. The adversary has no way to find these secret matrices other exhaustive search, which can be avoided by setting k'' to be large enough. Therefore, when the verifying party is using small size matrices H' and Y' for verification, for example when H' is 32×1760 and Y' is 32×440 (instead of 242×1760 and 242×440, receptively) for 128-bit security, although it easy to solve the $(n, k'', \omega) = (1760, 32, 484)$ instance with bit complexity 34, yet the adversary has no way for solving this instance because it is secret. Attempting to attack this mode of verification by solving all possible secret instances $(n, k'', \omega) = (1760, 32, 484)$ is unfeasible. Thus, we estimate time complexity of attacking light verification key mode (LVK Mode) of the signature scheme as

$$\binom{k'}{k''} \times T_{(n,k'',\omega)},$$

where $T_{(n,k'',\omega)}$ is time of solving the syndrome decoding instance $(H', \omega, Y'\bar{\mathbf{h}})$ which depends on the parameters n, k'', and ω.

We see the difference in size of the public key in the standard mode (Table IV) and size of the secret verification key in the LVK mode (Table VIII), which lead not only to smaller memory consumption in the later mode, but also to faster verification algorithm. Note that also the verification key must be secret only before its use, but it does not require to be kept for long time, a new secret verification key can always be selected from the original public key. However, small secret long-term verification key suits some applications.

IX. Light Signing Key (LSK) Mode

In this mode we minimize the size of the private key $sk = (K, P)$ by choosing the matrix $M = FGF^{-1}$ of equation (1) such that F is random permutation matrix which require order of k memory space instead of k^2. Thence for optimal memory usage we do not store the matrix M and its powers directly as $k \times k$ matrices. Instead, we store only its structure. The permutation matrix F is stored as $k-$tuple permutation; for ranges of k in this work, the $k-$tuple permutation representing F can be stored in $2k$ bytes. Let $l' = \frac{l}{2}$. The involutory matrix D is encoded as $l'-$bit string in which each bit b represents matrix D_b, $b \in \{0, 1\}$. The matrix C will be stored directly as an $m \times m$ matrix (though characteristic polynomials of the matrices C_i can be used to represent the matrix C, but for now let us assume that C is stored directly as a square matrix; it is anyway small matrix compared to D). Thus, the matrix M (and also its inverse and powers) is stored in the memory as: $2k$ bytes (size of permutation F) plus l' bits (size of the encoding of D) plus m^2 bits (size of C). That is, M can be stored in $(16k+l'+m^2)$ bits rather than k^2 bits. Note that from Equation (1) $m = k - l$, and note also that $l = \delta k = 0.9k$. Thus, matrix M can be stored in

$$(16k + l' + (0.10k)^2) \text{ bits},$$

TABLE V
BIT COMPLEXITY ESTIMATION FOR SOLVING THE UNDERLYING SYNDROME DECODING INSTANCE OF THE PARAMETER SET SSD1760.

Parameters: $\gamma = 4$, $\delta = 0.9$, $[n, k', d] = [1760, 242, 502]$, $\omega = 484$, $\mu = 11$						
SD Instance	$(n, k', \omega - \mu)$		(n, k', ω)		$(n, k', d + \mu)$	
Algorithm	Time	Memory	Time	Memory	Time	Memory
Prange [12]	148.5	21.6	151.9	21.6	152.1	21.6
Stern [13]	129.9	29.9	133.2	29.9	133.3	34.8
Dumer [14]	129.7	35.3	133.0	35.3	133.0	40.1
Ball Collision [15]	129.9	34.8	133.2	34.8	133.3	34.8
BJMM (MMT) [16]	127.8	53.3	131.1	53.3	131.4	53.3
BJMM-pdw [17]	128.9	50.3	132.1	50.3	132.2	41.6
May-Ozerov [18]	128.0	62.1	131.3	62.1	131.5	66.3
Both-May [19]	129.9	33.9	133.2	33.9	133.5	33.9

TABLE VI
BIT COMPLEXITY ESTIMATION FOR SOLVING THE UNDERLYING SYNDROME DECODING INSTANCE OF THE PARAMETER SET SSD2744.

Parameters: $\gamma = 4$, $\delta = 0.9$, $[n, k', d] = [2744, 337, 782]$, $\omega = 754$, $\mu = 11$						
SD Instance	$(n, k', \omega - \mu)$		(n, k', ω)		$(n, k', d + \mu)$	
Algorithm	Time	Memory	Time	Memory	Time	Memory
Prange	218.2	22.9	221.8	22.9	225.5	22.9
Stern	197.1	43.2	200.7	48.1	204.3	48.1
Dumer	196.7	48.9	200.3	48.9	203.9	48.9
Ball Collision	197.1	43.2	200.7	48.1	204.3	48.1
BJMM (MMT)	191.5	94.9	195.0	94.9	198.5	104.2
BJMM-pdw	193.8	68.0	197.4	72.0	201.0	72.0
May-Ozerov	191.9	105.9	195.6	101.9	199.4	98.3
Both-May	195.8	63.4	199.4	63.4	203.3	66.4

TABLE VII
BIT COMPLEXITY ESTIMATION FOR SOLVING THE UNDERLYING SYNDROME DECODING INSTANCE OF THE PARAMETER SET SSD3728.

Parameters: $\gamma = 4$, $\delta = 0.9$, $[n, k', d] = [3728, 512, 1063]$, $\omega = 1025$, $\mu = 12$						
SD Instance	$(n, k', \omega - \mu)$		(n, k', ω)		$(n, k', d + \mu)$	
Algorithm	Time	Memory	Time	Memory	Time	Memory
Prange	287.9	23.7	291.5	23.7	298.3	23.7
Stern	264.6	61.4	268.2	61.4	274.9	61.4
Dumer	264.5	51.9	268.1	51.9	274.9	51.9
Ball Collision	264.6	61.4	268.2	61.4	274.9	61.4
BJMM (MMT)	255.4	121.8	259.0	121.6	265.9	121.6
BJMM-pdw	258.5	93.1	262.2	97.1	269.0	102.3
May-Ozerov	256.9	105.1	260.6	105.1	267.8	105.1
Both-May	261.6	70.8	265.3	70.8	272.5	70.8

TABLE VIII
SIZES OF THE KEYS AND THE SIGNATURE FOR DIFFERENT SECURITY LEVELS OF THE LVK-*SSD* MODE.

Security Level	128-bits	192-bits	256-bits
Parameter Set	SSD1760	SSD2744	SSD3728
LVK Size (KB)	9	17	27.5
LSK Size (KB)	4.5	7.5	10.5
(Standard) Private Key Size (KB)	27	63	113
Signature Size (Bytes)	220	343	466

TABLE IX
PARAMETER SETS FOR LVK-SSD MODE

Parameter set	LVK-SSD1760	LVK-SSD2744	LVK-SSD3728
γ	4	4	4
δ	0.9	0.9	0.9
$[n, k', d]$	$[1760, 242, 502]$	$[2744, 337, 782]$	$[3728, 512, 1063]$
ω	484	754	1025
μ	11	12	12
k	440	686	932
k''	32	40	48
$\log \binom{k'}{k''}$	132	173	225

instead of k^2 bits as in the standard mode. When performing matrix exponentiation on $M = FGF^{-1}$, the matrix multiplication can be performed only on the matrix C, since D is involutory matrix with $D^2 = I_l$. In Subsection V-4 we see that the matrix K is represented by M^{-1} and the secret number α_1. In this mode the matrix M^{-1} has size $(16k + l' + (0.10k)^2)$ bits. Therefore, size of the private key $sk = (K, P)$ in the this mode is sum size of M^{-1} plus size of the permutation P plus size of α_1, that is $(16k+l'+(0.10k)^2)$ plus $16n$ plus $0.10k$ bits, respectively. Hence, size of the private key $sk = (K, P)$ in the LSK mode is

$$(16n) + (16k + l' + (0.10k)^2 + 0.10k) \text{ bits.}$$

(See Table X.)

TABLE X
SIZES OF THE KEYS AND THE SIGNATURE FOR DIFFERENT SECURITY LEVELS OF THE *LSK-SSD* MODE.

Security Level	128-bits	192-bits	256-bits
Parameter Set	SSD1760	SSD2744	SSD3728
Public Key Size (KB)	65	158	291.5
LSK Size (KB)	4.5	7.5	10.5
Signature Size (Bytes)	220	343	466

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[2] T. Chou, C. Cid, S. UiB, J. Gilcher, T. Lange, V. Maram, R. Misoczki, R. Niederhagen, K. Paterson, and E. Persichetti, "Classic mceliece: conservative code-based cryptography, 10 october 2020," 2020.

[3] N. Aragon, P. S. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor *et al.*, "Bike: bit flipping key encapsulation," 2017.

[4] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and I. Bourges, "Hamming quasi-cyclic (hqc)," *NIST PQC Round*, vol. 2, no. 4, p. 13, 2018.

[5] K. Pietrzak, "Cryptography from learning parity with noise," in *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 2012, pp. 99–114.

[6] NIST, "Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process," no. October, pp. 1–23, 2022. [Online]. Available: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf

[7] A. Esser and E. Bellini, "Syndrome decoding estimator," in *IACR International Conference on Public-Key Cryptography*. Springer, 2022, pp. 112–141.

[8] B. Aslan and M. T. Sakallı, "Algebraic construction of cryptographically good binary linear transformations," *Security and Communication Networks*, vol. 7, no. 1, pp. 53–63, 2014.

[9] A. M. A. Rushdi and F. A. M. Ghaleb, "On self-inverse binary matrices over the binary galois field," *Journal of Mathematics and Statistics*, vol. 9, no. 3, pp. 238–248, 2013.

[10] Guruswami, Venkatesh, Rudra, Atri, and Sudan, Madhu, "Essentials of Coding Theory," *Book*, 2022. [Online]. Available: https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/

[11] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on computing*, vol. 17, no. 2, pp. 281–308, 1988.

[12] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[13] J. Stern, "A method for finding codewords of small weight," in *International colloquium on coding theory and applications*. Springer, 1988, pp. 106–113.

[14] I. Dumer, "On minimum distance decoding of linear codes," in *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*, 1991, pp. 50–52.

[15] D. J. Bernstein, T. Lange, and C. Peters, "Smaller decoding exponents: ball-collision decoding," in *Annual Cryptology Conference*. Springer, 2011, pp. 743–760.

[16] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $\mathcal{O}(2^{0.054n})$," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 107–124.

[17] A. Becker, A. Joux, A. May, and A. Meurer, "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*. Springer, 2012, pp. 520–536.

[18] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 203–228.

[19] L. Both and A. May, "Decoding linear codes with high error rate and its impact for lpn security," in *International Conference on Post-Quantum Cryptography*. Springer, 2018, pp. 25–46.