Post-quantum Key Encapsulation Mechanism
EDON-$\mathcal{K}$

November 2017

Principal submitter

- Danilo Gligoroski, Norwegian University of Science and Technology (NTNU)
- E-mail address (preferred): `danilog@ntnu.no`
- Telephone: +47-735-94-616
- Postal address:
  - Danilo Gligoroski,
  - Department of Information Security and Communication Technologies,
  - Norwegian University of Science and Technology (NTNU),
  - O.S. Bragstass plass 2A,
  - 7034 Trondheim,
  - Norway

Auxiliary submitter:

- Kristian Gjøsteen, Norwegian University of Science and Technology (NTNU)
- E-mail address (preferred): `kristianj@ntnu.no`
- Telephone: +47-735-50-242
- Postal address:
  - Kristian Gjøsteen,
  - Department of Mathematics,
  - Norwegian University of Science and Technology (NTNU),
  - 7491 Trondheim,
  - Norway

Inventors/developers: The inventor/developer of this submission is the same as the principal submitter. Relevant prior work is credited below where appropriate.

Owner: Same as the principal submitter.

Signature: ×. See also printed version of "Statement by Each Submitter".

Document generated with the help of `pqskeleton` version 20170923.

Contents

1 Introduction

This documentation has been prepared in a response to the call for post-quantum cryptography standardization, issued by the National Institute of Standards and Technology (NIST) [12]. It describes in details the *key encapsulation mechanism* EDON-$\mathcal{K}$ which is submitted as a candidate for quantum-resistant *key establishment*.

EDON-$\mathcal{K}$ is a key encapsulation mechanism that support the establishment of shared secrets of length 256 and 384 bits. This documentation proposes parameters that provide a conjectured cryptographic strength for the Category 1 and Category 3 [12].

1.1 General properties and a global view of EDON-$\mathcal{K}$

EDON-$\mathcal{K}$ is a *key-establishment scheme*. Due to the relatively small public keys and their fast generation it can be used as *an ephemeral only key-establishment scheme*. Due to its IND-CCA2 security property, it can also store and reuse exchanged public keys. It security relies on the following assumptions:

1. The standardized hash functions SHA-256 and SHA-384 [13] act as one-way i.e. preimage resistant functions.
2. Finding a randomly generated 128-bit or 192-bit value, takes approximately $O(2^{128})$ and $O(2^{192})$ search and evaluate operations.
3. A new FINITE FIELD VECTOR SUBSET RATIO PROBLEM is NP-Hard problem, and as a multidimensional variant of the SUBSET-SUMS PROBLEM is a much harder problem.

The functionality of EDON-$\mathcal{K}$ follows the general key encapsulation mechanism (KEM):

KeyGen: Bob produces $(\mathbf{PubMat}_B, \mathbf{PrivMat}_B)$ key pair;

Encapsulate: Alice generates a shared secret *SharedSecret* and a ciphertext *Ciphertext* that "encapsulates" *SharedSecret*:

$$Ciphertext = Encapsulate(\mathbf{PubMat}_B, SharedSecret);$$

Decapsulate: Bob regenerates the SharedSecret *SharedSecret* from the ciphertext:

$$SharedSecret = Decapsulate(\mathbf{PrivMat}_B, Ciphertext).$$

In EDON-$\mathcal{K}$ the *SharedSecret* is long either 256 or 384 bits.

2 General algorithm specification (part of 2.B.1)

2.1 Parameters, variables and constants

The following parameters and variables are used in the specification of EDON-$\mathcal{K}$:

$sec = 128, 192,$	The security level of a concrete instance of the key establishment scheme.
EDON-$\mathcal{K}sec$	An EDON-$\mathcal{K}$ key establishment scheme with conjectured security that corresponds to at least *sec* classical bits. More concretely the instance EDON-$\mathcal{K}$128 corresponds to the Category 1 and EDON-$\mathcal{K}$192 to the Category 3 [12].
$\mathbb{F}_q,\ q \in \{2^{128},\ 2^{192}\}$	Finite field with $q = 2^{128}$, i.e. $q = 2^{192}$ elements.
$\mathbf{G} = [g_{i,j}]_{K \times N}$	A matrix with dimensions $K \times N$ with elements from a set $\mathcal{G}_{priv}$.
$\mathcal{G}_{base}$	A set of ν randomly chosen and different elements from $\mathbb{F}_q$, $\mathcal{G}_{base} = \{g_0, g_1, \ldots, g_{\nu-1}\}$.
$\mathbf{g}_{base}$	A ν-dim vector of $\mathcal{G}_{base}$ elements: $\mathbf{g}_{base} = (g_0, g_1, \ldots, g_{\nu-1})$.
$\mathcal{G}_{priv}$	A binary linear span of the set $\mathcal{G}_{base}$ i.e. $\mathcal{G}_{priv} = \{g \mid g = \mathbf{g}_{base} \cdot \mathbf{b},\ \ \mathbf{b} \in \{0,1\}^{\nu}\}$, where the operation $\cdot$ is a dot product.
$\mathbf{PubMat} = [p_{i,j}]_{K \times N}$	A matrix with dimensions $K \times N$ with elements from a set $\mathcal{P}_{pub}$.
PublicKey	An encoding of the matrix **PubMat** with a compression algorithm `CompressPublic`. Thus: $$PublicKey = \texttt{CompressPublic}(\mathbf{PubMat}).$$
$\mathcal{P}_{base}$	A set of 2ν elements from $\mathbb{F}_q$, $\mathcal{P}_{base} = \{cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}\}$, where $c, d \in \mathbb{F}_q$.
$\mathbf{p}_{base}$	A 2ν-dim vector of $\mathcal{P}_{base}$ elements: $\mathbf{p}_{base} = (cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1})$.
$\mathcal{P}_{pub}$	A binary linear span of the set $\mathcal{P}_{base}$ i.e. $\mathcal{P}_{pub} = \{p \mid p = \mathbf{p}_{base} \cdot \mathbf{b},\ \ \mathbf{b} \in \{0,1\}^{2\nu}\}$, where the operation $\cdot$ is a dot product.

$\mathbf{H}_{N\times R}$	A binary matrix which is called *annihilator matrix* where R is so called *projection dimension*.
$\mathbf{0} = [0]_{K\times R}$	A zero matrix with dimensions $K \times R$. The relation between $\mathbf{0}$, $\mathbf{G}$ and $\mathbf{H}$ is the following: $\mathbf{0} = \mathbf{G} \cdot \mathbf{H}$.
$\mathbf{P}_{a,b} = [p_{i,j}]_{N\times N}$	A non-singular, quasi-binary and quasi-orthogonal matrix with dimensions $N \times N$. Quasi-binary means that it has only two elements a and b i.e. $p_{i,j} \in \{a, b\}$. For elements a and b there are two corresponding elements c and d used in the construction of the set $\mathcal{P}_{base}$. Quasi-orthogonal means that its inverse matrix can be obtained by simple transposition of the original matrix, but an additional step should be performed: every appearance of a should be replaced by some value c, and every appearance of b should be replaced by d: $$\mathbf{P}_{a,b}^{-1} = (\mathbf{P}_{a,b})^T = \mathbf{P}_{c,d}.$$ The relations between a, b, c and d are the following: $$\begin{cases} c = & \dfrac{a}{a^2+b^2} \\ d = & \dfrac{b}{a^2+b^2} \end{cases}.$$
$\mathbf{PrivMat} = [q_{i,j}]_{N\times R}$	A private quasi-binary matrix with only two different elements $\{a, b\}$ from $\mathbb{F}_q$. The relations between **PrivMat**, **PubMat**, $\mathbf{G}$, $\mathbf{P}_{c,d}$, $\mathbf{P}_{a,b}$ and $\mathbf{H}$ are the following: $$\begin{cases} \mathbf{PubMat} = & \mathbf{G} \cdot \mathbf{P}_{c,d} \\ \mathbf{PrivMat} = & \mathbf{P}_{a,b} \cdot \mathbf{H} \end{cases}$$
PrivateKey	A 256-bit string generated uniformly at random.
SHA2: SHA-256 or SHA-384	NIST standardized cryptographic hash functions from SHA2 family. We use SHA-256 and SHA-384 with hash digest size of 256 and 384 bits [13].
$SHA2(s_0, s_1)$, $SHA2(s_0 \| \| s_1)$	In the context of arguments for the hash function $SHA2$ we use these two notations interchangeably.

2.2 Algorithms for KeyGen, Encapsulate and Decapsulate

```
KeyGen EDON-$\mathcal{K}$
Input: A security level $sec \in \{128, 192\}$
Output: $PublicKey$ and $PrivateKey$.

1. Choose the parameters $K$, $N$ and $v$ appropriate for the security level $sec$
2. $PrivateKey \xleftarrow{\$} \{0,1\}^{256}$
3. $\textsc{InitializeSeedExpander}(PrivateKey)$
4. $a, b, \mathbf{P}, \mathbf{H} \xleftarrow{\$\textsc{SeedExpander}} \{0,1\}^*$, where $a \neq b$ are two nonzero elements from $\mathbb{F}_q$, $\mathbf{P}$ is $N \times N$ binary orthogonal matrix, and the matrix $\mathbf{H}$ is a $N \times R$ binary matrix such that $\mathbf{H} = [\mathbf{HTop}||\mathbf{HBottom}]^T$, and where $\mathbf{HTop}$ is $(N-R) \times R$ matrix with all columns being of even Hamming weight, and $\mathbf{HBottom}$ is $R \times R$ binary orthogonal matrix
5. Compute $c$ and $d$ as follows: $\begin{cases} c = \dfrac{a}{a^2+b^2} \\ d = \dfrac{b}{a^2+b^2} \end{cases}$
6. Set $\mathbf{P}_{a,b}$ to be the quasi-binary quasi-orthogonal matrix obtained by $\mathbf{P}$ where every value of 0 is replaced by $a$ and every value of 1 is replaced by $b$. Set $\mathbf{P}_{c,d} = \mathbf{P}_{a,b}^{-1}$
7. $\mathbf{g}_{base} = (g_0, g_1, \ldots, g_{v-1}) \xleftarrow{\$} (\mathbb{F}_q)^v$, where $g_i \in \mathbb{F}_q$ are nonzero elements
8. Define $\mathcal{G}_{priv}$, to be a binary linear span of the elements of $\mathbf{g}_{base}$ as follows: $\mathcal{G}_{priv} = \{g \mid g = \mathbf{g}_{base} \cdot \mathbf{b},\ \mathbf{b} \in \{0,1\}^v\}$, where the operation $\cdot$ is a dot product
9. $\mathbf{GLeft}_{K \times (N-R)} \xleftarrow{\$} (\mathcal{G}_{priv})^{K \times (N-R)}$
10. $\mathbf{GRight}_{K \times R} = \mathbf{GLeft} \cdot \mathbf{HTop} \cdot \mathbf{HBottom}^{-1}$
11. Set $\mathbf{G} = [\mathbf{GLeft}||\mathbf{GRight}]$   (note that $\mathbf{0}_{K \times R} = \mathbf{G}_{K \times N} \cdot \mathbf{H}_{N \times R}$)
12. Set $\mathbf{PubMat}_{K \times N} = \mathbf{G} \cdot \mathbf{P}_{c,d}$
13. Set $PublicKey = \texttt{CompressPublic}(\mathbf{PubMat})$
14. Return $Publickey$ and $PrivateKey$.
```

Table 1: A generic description of the EDON-$\mathcal{K}$ key generation.

```
Encapsulate EDON-$\mathcal{K}$
Input: $PublicKey$
Output: A pair ($\mathbf{Ciphertext}, SharedSecret$).

1. Set $\mathbf{PubMat} = \texttt{DecompressPublic}(PublicKey)$
2. $\mathbf{M} = (m_0, m_1, \ldots, m_{K-1}) \xleftarrow{\$} (\mathbb{F}_q)^K$
3. Generate the vector $\mathbf{e}_{base} = (e_0, e_1, \ldots, e_{L-1})$, $e_i \in \mathbb{F}_q$ and $L$ is a public parameter for
   the corresponding security level of the scheme, as follows:
   (a) $(e_0, e_1) \xleftarrow{\$} (\mathbb{F}_q)^2$
   (b) $(e_{2i}, e_{2i+1}) \leftarrow SHA2(e_{2i-2}||e_{2i-1})$
       for $i \in \{1, \ldots, \frac{L}{2} - 1\}$
4. $\mathbf{B}_{L\times N} \xleftarrow{\$} \{0,1\}^{L\times N}$
5. $\mathbf{error} = \mathbf{e}_{base} \cdot \mathbf{B}$
6. $\mathbf{C} = \mathbf{M} \cdot \mathbf{PubMat} + \mathbf{error}$
7. $(s_0, s_1) = SHA2(e_{L-2}||e_{L-1})$
8. $SharedSecret = SHA2(s_0||s_1||SHA2(\mathbf{C}))$
9. $h = SHA2(s_1||s_0||SHA2(\mathbf{C}))$
10. Set $\mathbf{Ciphertext} = (\mathbf{C}, h)$
11. Return ($\mathbf{Ciphertext}, SharedSecret$).
```

Table 2: Description of the EDON-$\mathcal{K}$ key encapsulation

Decapsulate EDON-$\mathcal{K}$

```
Input: Ciphertext $= (\mathbf{C}, h)$, PrivateKey.
Output: SharedSecret or False.

1. InitializeSeedExpander(PrivateKey)
2. Recover $a, b, \mathbf{P}_{a,b}, \mathbf{H} \xleftarrow{\$_{\text{SeedExpander}}} \{0,1\}^*$
3. Compute $\mathbf{e}' = \mathbf{C} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} = (e'_1, \ldots, e'_R)$
4. Compute the following binary linear span set of the elements in $\mathbf{e}'$:
      $Candidates = \{\frac{s}{a+b} \mid \exists v \in \{0,1\}^R,\ s = v \cdot \mathbf{e}'\}$
5. If $|Candidates| > 2^{L+1}$ Return False.
6. If find a pair $(s_\mu, s_v) \in Candidates \times Candidates$ such that there exist $1 \le i \le \frac{L}{2}$ such
      that $(s_0, s_1) = \underbrace{SHA2(\ldots SHA2(s_\mu, s_v))}_{i}$ and $h = SHA2(s_1||s_0||SHA2(\mathbf{C}))$
      6.1 Set $SharedSecret = SHA2(s_0||s_1||SHA2(\mathbf{C}))$
      6.2 Return SharedSecret.
   else
      6.3 Return False.
```

Table 3: Description of the EDON-$\mathcal{K}$ key decapsulation

3 List of parameter sets (part of 2.B.1)

In Table 4 we give the parameters for our reference proposal for Edon-$\mathcal{K}$128 (**edonk128ref** row given in bold characters). That is also the name of the folder in the submission package for that proposal. The parameters for our reference proposal of Edon-$\mathcal{K}$128 are: $K = 16$, $N = 144$, $\nu = 8$, $R = 40$ and $L = 6$. We also give several alternative choices of parameters with their corresponding folder names in the submission package.

In the table, there are also several alternative choices of parameters that are without submission folder name. That means that there is no C code for these parameters in the submission package, but it would easy to produce them by simple copying and setting up the parameters in `*.h` files.

As we will see in Section 6, the expected strength for each parameter set of these parameters sets have expected strength that belong at least in Category 1 of the NIST call [12]. However, we followed the NIST recommendation to choose conservative parameters for the official reference proposal.

	Edon-K128 parameter space								
Submission folder name	K	N	ν	q GF(2^q)	R	L	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext Size (bytes)
	32	144	8	128	40	4	4896	32	2336
	32	96	8	128	40	4	3360	32	1568
	16	144	8	128	40	8	2576	32	2336
edonk128ref	**16**	**144**	**8**	**128**	**40**	**6**	**2576**	**32**	**2336**
edonk128K16N80nu8L6	16	80	8	128	40	6	1552	32	1312
	8	128	8	128	40	8	1288	32	2080
edonk128K08N72nu8L8	8	72	8	128	40	8	840	32	1184
	32	144	4	128	40	4	2448	32	2336
	32	128	4	128	40	4	2192	32	2080
edonk128K32N96nu4L4	32	96	4	128	40	4	1680	32	1568
	16	128	4	128	40	6	1160	32	2080
	16	96	4	128	40	6	904	32	1568
edonk128K16N80nu4L6	16	80	4	128	40	6	776	32	1312
	8	80	4	128	40	8	452	32	1312
	8	72	4	128	40	8	420	32	1184

Table 4: The parameter space for Edon-$\mathcal{K}$128.

In Table 5 we give the parameters for our reference proposal for Edon-$\mathcal{K}$192 (**edonk192ref** row given in bold characters). The parameters for our reference proposal of Edon-$\mathcal{K}$192 are: $K = 16$, $N = 112$, $\nu = 8$, $R = 40$ and $L = 8$.

	Edon-K192 parameter space								
Submission folder name	K	N	ν	q GF(2^q)	R	L	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext Size (bytes)
	44	144	8	192	40	4	6764	32	3504
	32	144	8	192	40	6	5024	32	3504
	28	128	8	192	40	6	3996	32	3120
edonk192ref	**16**	**112**	**8**	**192**	**40**	**8**	**2192**	**32**	**2736**
	16	144	8	192	40	8	2704	32	3504
edonk192K48N144nu4L4	48	144	4	192	40	4	3672	32	3504
edonk192K32N128nu4L6	32	128	4	192	40	6	2256	32	3120
	28	144	4	192	40	6	2222	32	3504
	24	144	4	192	40	8	1932	32	3504
	20	128	4	192	40	8	1482	32	3120
	16	128	4	192	40	8	1224	32	3120
edonk192K16N112nu4L8	16	112	4	192	40	8	1096	32	2736

Table 5: The parameter space for Edon-$\mathcal{K}$192.

For making an easy mind-map between the mathematical notations for the parameters K, N, ν, R and L (and **PrivMat** matrix) in this documentation and our C code implementation, in Table 6 we give the names of variables that correspond to these parameters.

Parameter name	C code variable name
K	`NrRows`
N	`NrColumns`
ν	`BinarySpan`
R	`ProjectionDim`
L	`ErrorBasisDimension`
PrivMat	`ShortenedPermutation`

Table 6: Translation table for the names of the parameters in this documentation and the names of variables in our C implementation

4 Design rationale and goals (part of 2.B.1)

We explain the design rationale by first listing the design goals **G1**, …, **G5** that we set for EDON-$\mathcal{K}$:

G1. To design a code-based public-key key encapsulation scheme. That means the public key to have a form of a $K \times N$ matrix **PubMat** such that the encoding and decoding are performed in a McEliece style:

$$\mathbf{C} = \mathbf{m} \cdot \mathbf{PubMat} + \mathbf{error}.$$

G2. To be able to generate public and private keys very efficiently.

G3. **PubMat** to have relatively small size (from 0.5 KB to 4 KB).

G4. Information Set Decoding types of attacks to be ineffective for the scheme.

G5. The scheme to be IND-CCA2 secure by design, i.e. to need not any extra conversion method to achieve IND-CCA2 security.

4.1 EDON-$\mathcal{K}$ is a variant of McEliece public key scheme (G1.)

Let us very briefly recall how McEliece scheme [11] is designed.

The public key is a $K \times N$ binary matrix **PubMat**:

$$\mathbf{PubMat} = \mathbf{B} \cdot \mathbf{G} \cdot \mathbf{P},$$

where **B** is a $K \times K$ nonsingular binary matrix, **G** is a generator matrix of a Goppa code [8] that can correct up to t errors, and **P** is a $N \times N$ permutation matrix. The private key is the triplet of matrices $(\mathbf{B}, \mathbf{G}, \mathbf{P})$. The role of the matrices **B** and **P** is to hide the structure of the matrix **G**. Moreover, neither the matrix **P** nor its inverse $\mathbf{P}^{-1}$, as a permutation matrices, do not change the Hamming weight of the result, when they multiply vectors with of Hamming weight t. This can be shortly written as: if $HammingWeight(\mathbf{error}) = t$, then $HammingWeight(\mathbf{error} \cdot \mathbf{P}^{-1}) = t$.

The encryption of a binary vector **m** of length K is performed by choosing a binary error vector **error** of length N and Hamming weight t and by computing $\mathbf{C} = \mathbf{m} \cdot \mathbf{PubMat} + \mathbf{error}$.

In the decryption phase, first the vector $\mathbf{C}' = \mathbf{C} \cdot \mathbf{P}^{-1}$ is computed. Then the error correction algorithm `GoppaCorrect` for the Goppa code is applied to produce a vector $\mathbf{m}' = \texttt{GoppaCorrect}(\mathbf{C}')$. Finaly **m** is obtained by $\mathbf{m} = \mathbf{B}^{-1} \cdot \mathbf{m}'$.

Our modifications of the McEliece scheme are the following:

1. We construct **PubMat** over the finite fields $\mathbb{F}_q$. In particular, in this document $q \in \{2^{128},\ 2^{192}\}$.
2. We do not use the matrix **B**.
3. Instead of permutation matrix **P** we use another class of matrices: *quasi-binary quasi-orthogonal matrices*.
4. The error vectors used in the encryption phase are not restricted by the Hamming weight metric.

4.2 The mathematical structures for achieving the goal (G2.)

Definition 1. A square nonsingular matrix $\mathbf{P}$ is orthogonal if $\mathbf{P}^{-1} = \mathbf{P}^T$.

The practical benefits of using orthogonal matrices in coding theory come from the fact that the computation of inverse matrices is avoided, simply by usint the transpose of the orthogonal matrix. A pioneering work in the orthogonal circulant matrices over finite fields was done by MacWilliams [10]. That work was extended by Byrd and Vaughan in [4], and Zhang in [16] gave algorithms for constructing orthogonal circulant matrices for arbitrary dimensions N in any finite field.

In the construction of EDON-$\mathcal{K}$ we use one related class of matrices that we call *quasi-binary quasi-orthogonal matrices* defined as follows:

Definition 2. Let $\mathbf{P}$ be an $N \times N$ orthogonal binary matrix, and let $a \neq b$ are two nonzero elements from $\mathbb{F}_q$. We call the matrix $\mathbf{P}_{a,b}$ obtained from $\mathbf{P}$, a quasi-binary quasi-orthogonal matrix if every value of 0 is replaced by a and every value of 1 is replaced by b.

Apparently, the reason why we call the matrices quasi-binary is due to the fact that matrices have only two elements $a, b \in \mathbb{F}_q$, but those elements are not 0 and 1. On the other hand, the reasons why we call them quasi-orthogonal is due to the fact that their inverses are obtained by a simple transposition and replacement of a and b by two other values c and d as stated in the following theorem:

Theorem 1. For the inverse matrix $\mathbf{P}_{a,b}^{-1}$ it holds that $\mathbf{P}_{a,b}^{-1} = (\mathbf{P}_{a,b})_{c,d}^T$, i.e., every value of a in the transposition matrix $(\mathbf{P}_{a,b})^T$ is replaced by c and every value of b is replaced by d where

$$\begin{cases} c = & \dfrac{a}{a^2+b^2} \\ d = & \dfrac{b}{a^2+b^2} \end{cases}. \tag{1}$$

We write $\mathbf{P}_{a,b}^{-1} = \mathbf{P}_{c,d}$.

Example 1. Let us work in this example in the finite field $\mathbb{F}_{16}$ with an irreducible polynomial: $x^4 + x^3 + 1$. Next, instead of representing the elements $a \in \mathbb{F}_{16}$ as polynomials $a = a_3x^3 + a_2x^2 + a_1x + a_0$, where $a_i \in \{0, 1\}$, let us represent the elements with the corresponding integer values from their binary representation. For example, if $a = 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x + 1$, then we will write $a = (0, 1, 1, 1)_2 = 7$.

Let us have the following binary matrix:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

It is easy to check that $\mathbf{P} \cdot \mathbf{P}^T = I$, i.e. that $\mathbf{P}$ is an orthogonal binary matrix. Let us choose the following values: $a = 7$, $b = 13$ and their corresponding values $c = 4$ and $d = 15$. Then the two quasi-binary and quasi-orthogonal matrices are:

$$\mathbf{P}_{7,13} = \begin{pmatrix} 7 & 13 & 7 & 13 & 13 & 7 & 7 & 7 \\ 7 & 7 & 13 & 7 & 7 & 13 & 7 & 13 \\ 7 & 7 & 7 & 7 & 7 & 13 & 13 & 13 \\ 7 & 7 & 13 & 7 & 7 & 13 & 13 & 7 \\ 7 & 7 & 13 & 7 & 7 & 7 & 13 & 13 \\ 13 & 13 & 7 & 7 & 13 & 7 & 7 & 7 \\ 13 & 7 & 7 & 13 & 13 & 7 & 7 & 7 \\ 13 & 13 & 7 & 13 & 7 & 7 & 7 & 7 \end{pmatrix} \quad \mathbf{P}_{4,15} = \begin{pmatrix} 4 & 4 & 4 & 4 & 4 & 15 & 15 & 15 \\ 15 & 4 & 4 & 4 & 4 & 15 & 4 & 15 \\ 4 & 15 & 4 & 15 & 15 & 4 & 4 & 4 \\ 15 & 4 & 4 & 4 & 4 & 4 & 15 & 15 \\ 15 & 4 & 4 & 4 & 4 & 15 & 15 & 4 \\ 4 & 15 & 15 & 15 & 4 & 4 & 4 & 4 \\ 4 & 4 & 15 & 15 & 15 & 4 & 4 & 4 \\ 4 & 15 & 15 & 4 & 15 & 4 & 4 & 4 \end{pmatrix}.$$

Next, in our attempt to achieve the goal of efficiency, instead of using Zhang's algorithms for construction of orthogonal matrices with arbitrary dimensions N we use an approach of constructing incidence matrices from Latin Rectangles. That type of matrices we constructed for the hash function EDON-$\mathcal{R}$ [6] and for the MQQ-SIG multivariate signature scheme [7].

Without going deeper in the Combinatorics, we give here the basic definitions and basic propositions for Latin Rectangles and incidence matrices. Proofs of these basic properties can be found for example in [14]

Definition 3. A $k \times n$ Latin Rectangle $L = [l_{i,j}]_{k \times n}$ is a $k \times n$ array (where $k \leq n$) in which every row $R_0, \ldots, R_{k-1}$ is a permutation of an n-element set $X = \{0, 1, \ldots, n-1\}$, and the elements in each column $C_0, C_1, \ldots, C_{n-1}$ appear at most once. Note here the zero-indexing style i.e. $i, j \in \{0, 1, \ldots, n-1\}$.

Definition 4. Let $L = [l_{i,j}]_{k \times n}$ be a Latin Rectangle, with columns $C_0, C_1, \ldots, C_{n-1}$. The incidence matrix of L is the $n \times n$ binary matrix $\mathbf{M} = (m_{i,j})$ defined by the rule

$$m_{i,j} = \begin{cases} 1, & \text{if } i \in C_j, \\ 0, & \text{if } i \notin C_j. \end{cases}$$

Example 2. Let $k = 3$ and $n = 8$, and let $L = [l_{i,j}]_{3\times 8} = \begin{bmatrix} 6 & 5 & 4 & 3 & 1 & 7 & 0 & 2 \\ 4 & 3 & 1 & 7 & 0 & 2 & 6 & 5 \\ 1 & 7 & 0 & 2 & 6 & 5 & 4 & 3 \end{bmatrix}$. Then the corresponding incidence matrix is: $\mathbf{M} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$.

Proposition 1. The incidence matrix $M = (m_{i,j})$ of any Latin Rectangle with dimensions $k \times n$ is balanced matrix with k ones in each row and each column.

Proposition 2. Let $M = (m_{i,j})$ be the incidence matrix of a Latin Rectangle with dimensions $k \times n$ and even n. If M is nonsingular, then k is odd.

Definition 5. Let R_0 be a permutation of the n-element set $X = \{0, 1, \ldots, n-1\}$. Let $gcd(n, rot) = 1$ and let $R_i = RotateLeft(R_0, i \times rot)$ are obtained with left rotation of the initial permutation R_0 by $i \times rot$ positions. Then the Latin Rectangle $L = [l_{i,j}]_{k\times n}$ with rows $R_0, \ldots, R_{k-1}$ is called a Cyclic Latin Rectangle.

In the Example 2 the initial permutation is $R_0 = \{6, 5, 4, 3, 1, 7, 0, 2\}$ and R_1 and R_2 are obtained with its rotation by 2 elements to the left.

Observation: For certain values of even n, there are values of rot and k such that the incidence matrices that correspond to the cyclic Latin Rectangles defined by Definition 5 are orthogonal.

In our pursue for the efficiency goal **G2** we were interested in cyclic Latin Rectangles that give incidence matrices that are orthogonal, with as small as possible number of rows k. By running a simple Sagemath script for even n in the range $\{8, 10, 12, \ldots, 256\}$, we constructed the Table 7 for the values of (n, k, rot). The entries in that table should be interpreted as follows: If the triplet (n, k, rot) is present in the table, then there is an efficient procedure for generating orthogonal binary matrices of size $n \times n$ from a cyclic Latin Rectangle with k rows, by generating one random permutation R_0 with n elements $\{0, 1, \ldots, n-1\}$, and by producing $k-1$ subsequent rows with rotating R_0 by rot positions to the left.

Next we use the following property of orthogonal matrices:

(8,3,2)	(12,3,3)	(16,3,4)	(18,5,3)	(20,3,5)	(24,3,6)	(28,3,7)	(30,5,5)
(32,3,8)	(36,3,9)	(40,3,10)	(42,5,7)	(44,3,11)	(48,3,12)	(50,9,5)	(52,3,13)
(54,5,9)	(56,3,14)	(60,3,15)	(64,3,16)	(66,5,11)	(68,3,17)	(70,9,7)	(72,3,18)
(76,3,19)	(78,5,13)	(80,3,20)	(84,3,21)	(88,3,22)	(90,5,15)	(92,3,23)	(96,3,24)
(98,13,7)	(100,3,25)	(102,5,17)	(104,3,26)	(108,3,27)	(110,9,11)	(112,3,28)	(114,5,19)
(116,3,29)	(120,3,30)	(124,3,31)	(126,5,21)	(128,3,32)	(130,9,13)	(132,3,33)	(136,3,34)
(138,5,23)	(140,3,35)	(144,3,36)	(148,3,37)	(150,5,25)	(152,3,38)	(154,13,11)	(156,3,39)
(160,3,40)	(162,5,27)	(164,3,41)	(168,3,42)	(170,9,17)	(172,3,43)	(174,5,29)	(176,3,44)
(180,3,45)	(182,13,13)	(184,3,46)	(186,5,31)	(188,3,47)	(190,9,19)	(192,3,48)	(196,3,49)
(198,5,33)	(200,3,50)	(204,3,51)	(208,3,52)	(210,5,35)	(212,3,53)	(216,3,54)	(220,3,55)
(222,5,37)	(224,3,56)	(228,3,57)	(230,9,23)	(232,3,58)	(234,5,39)	(236,3,59)	(238,13,17)
(240,3,60)	(242,21,11)	(244,3,61)	(246,5,41)	(248,3,62)	(250,9,25)	(252,3,63)	(256,3,64)

Table 7: Triplets (n, k, rot) for which our routine can produce orthogonal binary matrices.

Proposition 3. If A and B are two orthogonal matrices then their product $A \cdot B$ is an orthogonal matrix.

If we combine Proposition 1 and Proposition 3 it give us a way how to construct random looking binary orthogonal matrices. For a given set of parameters (n, k, rot) from Table 7 we produce several random instances of binary orthogonal matrices, and we multiply them all together to produce a final result.

Example 3. The orthogonal matrix $\mathbf{P}$ from Example 1 is obtained as the following product $\mathbf{P} = \mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3$ from the following Latin Rectangles and their corresponding incidence matrices:

$$L_1 = [l_{i,j}]_{3\times 8} = \begin{bmatrix} 6 & 5 & 4 & 3 & 1 & 7 & 0 & 2 \\ 4 & 3 & 1 & 7 & 0 & 2 & 6 & 5 \\ 1 & 7 & 0 & 2 & 6 & 5 & 4 & 3 \end{bmatrix} \text{ and } \mathbf{M}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$L_2 = [l_{i,j}]_{3\times 8} = \begin{bmatrix} 3 & 6 & 1 & 7 & 4 & 2 & 0 & 5 \\ 1 & 7 & 4 & 2 & 0 & 5 & 3 & 6 \\ 4 & 2 & 0 & 5 & 3 & 6 & 1 & 7 \end{bmatrix} \text{ and } \mathbf{M}_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$L_3 = [l_{i,j}]_{3\times 8} = \begin{bmatrix} 4 & 2 & 5 & 6 & 3 & 1 & 0 & 7 \\ 5 & 6 & 3 & 1 & 0 & 7 & 4 & 2 \\ 3 & 1 & 0 & 7 & 4 & 2 & 5 & 6 \end{bmatrix} \text{ and } \mathbf{M}_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The only remaining part is to find a way how to avoid a direct classical matrix multiplication procedure, but still to construct an orthogonal matrix that is a product of two orthogonal matrices. We are achieving that goal with the following Proposition:

Proposition 4. Let $L_1 = [l_{i,j}]_{k\times n}$ with its corresponding columns $C_0^{(1)}, C_1^{(1)}, \ldots, C_{n-1}^{(1)}$ and $L_2 = [l_{i,j}]_{k\times n}$ with its corresponding columns $C_0^{(2)}, C_1^{(2)}, \ldots, C_{n-1}^{(2)}$ are two Latin Rectangles, and let $\mathbf{M}_1$ and $\mathbf{M}_2$ are their corresponding incidence matrices.

Let define n initial sets $D_i^{(0)} = \{i\}, i \in \{0, \ldots, n-1\}$, let the operator $\bigoplus$ denote exclusive union of sets and let us define $D_i^{(j)}$ with the following recursive relations:

$$D_i^{(j)} = \bigoplus_{l \in C_i^{(j)}} D_l^{(j-1)}, \text{ for } i = 0, 1, \ldots n-1, \text{ and for } j = 1, 2 \tag{2}$$

Then, the sets $D_i^{(2)}, i \in \{0, \ldots, n-1\}$ are the support sets for the columns of the product matrix: $\mathbf{P} = \mathbf{M}_1 \cdot \mathbf{M}_2$.

Proof. (Sketch) The correctness of relation (2) comes from Definition 4 for incidence matrices and the standard definition of matrix multiplication applied to matrices $\mathbf{M}_1$ and $\mathbf{M}_2$. ☐

We summarize all definitions, propositions and the observation in this subsection in the algorithm `RandomOrthogonalBinaryMatrix`(n) for generation of binary orthogonal matrices of size $n \times n$, given in Table 8. Note that we chose 6 iterations in Step 3 of the algorithm `RandomOrthogonalBinaryMatrix`(n). That is an arbitrary value obtained by our concrete experiments, that achieves the average Hamming weight of the columns and rows in the produced orthogonal matrix to be distributed around $\frac{n}{2}$. Further research is needed to analyze the possibility this number to be less than 6, without jeopardizing the security of the overall scheme.

4.3 The mathematical structures for achieving the goal (G3.)

In order to have a small public matrix **PubMat** we chose the elements of the $K \times N$ matrix $\mathbf{G}$ from a binary linear span of a small random subset of elements in $\mathbb{F}_q$.

```
RandomOrthogonalBinaryMatrix
Input: $n$,
Output: A random orthogonal $n \times n$ binary matrix $\mathbf{P}$.

1. For the given $n$ find the corresponding triplet $(n, k, rot)$ from the Table 7. If the triplet is not present, Return Error
2. Initialize $n$ sets $D_i^{(0)} = \{i\}, i \in \{0, \ldots, n-1\}$ as in Proposition 4
3. For $j = 1$ to 6 do
     • Generate a random cyclic Latin Rectangle $L_j$ of size $k \times n$ with columns $C_0^{(j)}, C_1^{(j)}, \ldots, C_{n-1}^{(j)}$
     • $D_i^{(j)} = \bigoplus_{l \in C_i^{(j)}} D_l^{(j-1)}, \ \text{for } i = 0, 1, \ldots n-1$
4. Set a binary matrix $\mathbf{P}$ of size $n \times n$ for which the sets $D_i^{(6)}, i \in \{0, \ldots, n-1\}$ are the support sets for its columns
5. Return $\mathbf{P}$.
```

Table 8: The algorithm for fast construction of orthogonal binary matrices of size $n \times n$.

Definition 6. Let $\mathcal{G}_{base}$ be a set of ν randomly chosen and different elements from $\mathbb{F}_q$, $\mathcal{G}_{base} = \{g_0, g_1, \ldots, g_{\nu-1}\}$, and let $\mathbf{g}_{base}$ be a ν-dim vector of $\mathcal{G}_{base}$ elements: $\mathbf{g}_{base} = (g_0, g_1, \ldots, g_{\nu-1})$. A binary linear span of the set $\mathcal{G}_{base}$ is the following set: $\mathcal{G}_{priv} = \{g \mid g = \mathbf{g}_{base} \cdot \mathbf{b}, \ \mathbf{b} \in \{0,1\}^{\nu}\}$, where the operation $\cdot$ is a dot product.

Definition 7. The elements of the matrix $\mathbf{G}$ are chosen from the set $\mathcal{G}_{priv}$. The elements a, b, c, d defined in Definition 2 and Theorem 1 do not belong to the set $\mathcal{G}_{base}$.

Although the values for ν can be arbitrary $\nu \geq 2$, in this proposal for EDON-$\mathcal{K}$ for the NIST post-quantum standardization, we decided the values for the parameter ν to be either 4 or 8. As we will see further, that gives a nice data structure for the elements of the public key that can be described with half byte or with one byte i.e., with 4 or 8 bits.

Definition 8. Let define an $N \times R$ binary matrix $\mathbf{H}$ such that $\mathbf{H} = \begin{bmatrix} \mathbf{HTop} \\ \mathbf{HBottom} \end{bmatrix}$, and where $\mathbf{HTop}$ is $(N-R) \times R$ matrix with all columns being of even Hamming weight, and $\mathbf{HBottom}$ is $R \times R$ binary orthogonal matrix defined in Section 4.2.

It is a simple exercise to show the correctness of the following Proposition:

Proposition 5. Let $\mathbf{G} = [\mathbf{GLeft} || \mathbf{GRight}]$ be a $K \times N$ matrix, where

$$\mathbf{GRight} = \mathbf{GLeft} \cdot \mathbf{HTop} \cdot \mathbf{HBottom}^{-1}.$$

Then, $\mathbf{G} \cdot \mathbf{H} = \mathbf{0}$.

Definition 9. Let the matrices $\mathbf{G}$ and $\mathbf{H}$ be defined with Definition 6, 7, 8 and Proposition 5, and let the matrices $\mathbf{P}_{a,b}$ and $\mathbf{P}_{c,d}$ be defined with Definition 2 and Theorem 1. We define the matrices **PubMat** and **PrivMat** as:

$$\mathbf{PubMat} = \mathbf{G} \cdot \mathbf{P}_{c,d}, \tag{3}$$

and

$$\mathbf{PrivMat} = \mathbf{P}_{a,b} \cdot \mathbf{H}. \tag{4}$$

Having that $\mathbf{H}$ is annihilator matrix for $\mathbf{G}$ it directly follows that:

Corollary 1. The matrix **PrivMat** is annihilator matrix for the matrix **PubMat** i.e.,

$$\mathbf{0} = \mathbf{PubMat} \cdot \mathbf{PrivMat}.$$

□

Proposition 6. Every element of the matrix **PubMat** belongs to the binary linear span of the following set: $\mathcal{P}_{base} = \{cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}\}$.

Proof. Since the elements of $\mathbf{G}$ are from the binary linear span $\mathcal{G}_{priv} = \{g \mid g = \mathbf{g}_{base} \cdot \mathbf{b}, \ \ \mathbf{b} \in \{0,1\}^{\nu}\}$, and the elements of the matrix $\mathbf{P}_{c,d}$ are only c and d, it is clear that the product matrix **PubMat** has elements that are binary linear combinations of the elements from the set $\mathcal{P}_{base} = \{cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}\}$. □

Theorem 2. The linear binary span of the set $\mathcal{P}_{base} = \{cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}\}$ consists of $2^{2\nu}$ elements that can be partitioned in 2^{ν} coset classes.

Proof. Since the number of elements in $\mathcal{P}_{base}$ is 2ν, it is clear that the number of elements in its linear binary span is $2^{2\nu}$. Now, let us partition its linear binary span in the following way. Let us denote the vector $\mathbf{p}_{base} = (cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1})$. The basic coset $Coset_0$ is defined as

$$Coset_0 = \{\mathbf{p} \ \mid \ \mathbf{p} = \mathbf{p}_{base} \cdot \mathbf{b}, \text{where } \mathbf{b} = (\mathbf{b}_1, \mathbf{b}_1), \mathbf{b}_1 \in \{0,1\}^{\nu}\}.$$

Then, for any other $\mathbf{b}_2 \in \{0,1\}^{\nu}$ we define the cosets

$$Coset_{\mathbf{b}_2} = \{\mathbf{p} \ \mid \ \mathbf{p} = \mathbf{p}_0 + \mathbf{b}, \text{where } \mathbf{p}_0 \in Coset_0 \text{ and } \mathbf{b} = (\mathbf{b}_0, 0), \mathbf{b}_0 \in \{0,1\}^{\nu}\}.$$

□

Example 4. Let us take $\nu = 4$, and let us rename the elements of $\mathbf{p}_{base} = (cg_0, cg_1, cg_2, cg_3, dg_0, dg_1, dg_2, dg_3)$ as $\mathbf{p}_{base} = (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$. Then, all $2^{2 \cdot 4} = 2^8 = 256$ elements of the binary linear span can be partitioned in $2^4 = 16$ coset classes. We construct one such partitioning in a similar way as it is given in the proof of Theorem 2. Due to the large size of the table, we give it in two parts and in a landscape orientation on the next page.

	0	1	2	3	4	5	6	7
0	0	$p_1 + p_5$	$p_2 + p_6$	$p_1 + p_2 + p_5 + p_6$	$p_3 + p_7$	$p_1 + p_3 + p_5 + p_7$	$p_2 + p_3 + p_6 + p_7$	$p_1 + p_2 + p_3 + p_5 + p_6 + p_7$
1	p_1	p_5	$p_1 + p_2 + p_6$	$p_2 + p_5 + p_6$	$p_1 + p_3 + p_7$	$p_3 + p_5 + p_7$	$p_1 + p_2 + p_3 + p_6 + p_7$	$p_2 + p_3 + p_5 + p_6 + p_7$
2	p_2	p_6	$p_1 + p_2 + p_5$	$p_1 + p_5 + p_6$	$p_2 + p_3 + p_7$	$p_1 + p_2 + p_3 + p_5 + p_7$	$p_3 + p_6 + p_7$	$p_1 + p_3 + p_5 + p_6 + p_7$
3	p_3	p_7	$p_1 + p_3 + p_5$	$p_2 + p_3 + p_6$	$p_1 + p_2 + p_3 + p_5 + p_6$	$p_1 + p_5 + p_7$	$p_2 + p_6 + p_7$	$p_1 + p_2 + p_5 + p_6 + p_7$
4	p_4	p_8	$p_1 + p_4 + p_5$	$p_2 + p_4 + p_6$	$p_1 + p_2 + p_4 + p_5 + p_6$	$p_3 + p_4 + p_7$	$p_1 + p_3 + p_4 + p_5 + p_7$	$p_2 + p_3 + p_4 + p_6 + p_7$
5	$p_1 + p_2$	$p_2 + p_5$	$p_1 + p_6$	$p_5 + p_6$	$p_1 + p_2 + p_3 + p_7$	$p_2 + p_3 + p_5 + p_7$	$p_1 + p_3 + p_6 + p_7$	$p_3 + p_5 + p_6 + p_7$
6	$p_1 + p_3$	$p_3 + p_5$	$p_1 + p_2 + p_3 + p_6$	$p_2 + p_3 + p_5 + p_6$	$p_1 + p_7$	$p_5 + p_7$	$p_1 + p_2 + p_6 + p_7$	$p_2 + p_5 + p_6 + p_7$
7	$p_2 + p_3$	$p_1 + p_2 + p_3 + p_5$	$p_3 + p_6$	$p_1 + p_3 + p_5 + p_6$	$p_2 + p_7$	$p_1 + p_2 + p_5 + p_7$	$p_6 + p_7$	$p_1 + p_5 + p_6 + p_7$
8	$p_1 + p_2 + p_3$	$p_2 + p_3 + p_5$	$p_1 + p_3 + p_6$	$p_3 + p_5 + p_6$	$p_1 + p_2 + p_7$	$p_2 + p_5 + p_7$	$p_1 + p_6 + p_7$	$p_5 + p_6 + p_7$
9	$p_1 + p_4$	$p_4 + p_5$	$p_1 + p_2 + p_4 + p_6$	$p_2 + p_4 + p_5 + p_6$	$p_1 + p_3 + p_4 + p_7$	$p_3 + p_4 + p_5 + p_7$	$p_1 + p_2 + p_3 + p_4 + p_6 + p_7$	$p_2 + p_3 + p_4 + p_5 + p_6 + p_7$
10	$p_2 + p_4$	$p_1 + p_2 + p_4 + p_5$	$p_4 + p_6$	$p_1 + p_4 + p_5 + p_6$	$p_2 + p_3 + p_4 + p_7$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_7$	$p_3 + p_4 + p_6 + p_7$	$p_1 + p_3 + p_4 + p_5 + p_6 + p_7$
11	$p_1 + p_2 + p_4$	$p_2 + p_4 + p_5$	$p_1 + p_4 + p_6$	$p_4 + p_5 + p_6$	$p_1 + p_2 + p_3 + p_4 + p_7$	$p_2 + p_3 + p_4 + p_5 + p_7$	$p_1 + p_3 + p_4 + p_6 + p_7$	$p_3 + p_4 + p_5 + p_6 + p_7$
12	$p_3 + p_4$	$p_1 + p_3 + p_4 + p_5$	$p_2 + p_3 + p_4 + p_6$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_6$	$p_4 + p_7$	$p_1 + p_4 + p_5 + p_7$	$p_2 + p_4 + p_6 + p_7$	$p_1 + p_2 + p_4 + p_5 + p_6 + p_7$
13	$p_1 + p_3 + p_4$	$p_3 + p_4 + p_5$	$p_1 + p_2 + p_3 + p_4 + p_6$	$p_2 + p_3 + p_4 + p_5 + p_6$	$p_1 + p_4 + p_7$	$p_4 + p_5 + p_7$	$p_1 + p_2 + p_4 + p_6 + p_7$	$p_2 + p_4 + p_5 + p_6 + p_7$
14	$p_2 + p_3 + p_4$	$p_1 + p_2 + p_3 + p_4 + p_5$	$p_3 + p_4 + p_6$	$p_1 + p_3 + p_4 + p_5 + p_6$	$p_2 + p_4 + p_7$	$p_1 + p_2 + p_4 + p_5 + p_7$	$p_4 + p_6 + p_7$	$p_1 + p_4 + p_5 + p_6 + p_7$
15	$p_1 + p_2 + p_3 + p_4$	$p_2 + p_3 + p_4 + p_5$	$p_1 + p_3 + p_4 + p_6$	$p_3 + p_4 + p_5 + p_6$	$p_1 + p_2 + p_4 + p_7$	$p_2 + p_4 + p_5 + p_7$	$p_1 + p_4 + p_6 + p_7$	$p_4 + p_5 + p_6 + p_7$

	8	9	10	11	12	13	14	15
0	$p_4 + p_8$	$p_1 + p_4 + p_5 + p_8$	$p_2 + p_4 + p_6 + p_8$	$p_1 + p_2 + p_4 + p_5 + p_6 + p_8$	$p_3 + p_4 + p_7 + p_8$	$p_1 + p_3 + p_4 + p_5 + p_7 + p_8$	$p_2 + p_3 + p_4 + p_6 + p_7 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8$
1	$p_1 + p_4 + p_8$	$p_4 + p_5 + p_8$	$p_1 + p_2 + p_4 + p_6 + p_8$	$p_2 + p_4 + p_5 + p_6 + p_8$	$p_1 + p_3 + p_4 + p_7 + p_8$	$p_3 + p_4 + p_5 + p_7 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_6 + p_7 + p_8$	$p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8$
2	$p_2 + p_4 + p_8$	$p_1 + p_2 + p_4 + p_5 + p_8$	$p_4 + p_6 + p_8$	$p_1 + p_4 + p_5 + p_6 + p_8$	$p_2 + p_3 + p_4 + p_7 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_7 + p_8$	$p_3 + p_4 + p_6 + p_7 + p_8$	$p_1 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8$
3	$p_3 + p_4 + p_8$	$p_1 + p_3 + p_4 + p_5 + p_8$	$p_2 + p_3 + p_4 + p_6 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_8$	$p_4 + p_7 + p_8$	$p_1 + p_4 + p_5 + p_7 + p_8$	$p_2 + p_4 + p_6 + p_7 + p_8$	$p_1 + p_2 + p_4 + p_5 + p_6 + p_7 + p_8$
4	$p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7$	$p_1 + p_5 + p_8$	$p_2 + p_6 + p_8$	$p_1 + p_2 + p_5 + p_6 + p_8$	$p_3 + p_7 + p_8$	$p_1 + p_3 + p_5 + p_7 + p_8$	$p_2 + p_3 + p_6 + p_7 + p_8$	$p_1 + p_2 + p_3 + p_5 + p_6 + p_7 + p_8$
5	$p_1 + p_2 + p_4 + p_8$	$p_2 + p_4 + p_5 + p_8$	$p_1 + p_4 + p_6 + p_8$	$p_4 + p_5 + p_6 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_7 + p_8$	$p_2 + p_3 + p_4 + p_5 + p_7 + p_8$	$p_1 + p_3 + p_4 + p_6 + p_7 + p_8$	$p_3 + p_4 + p_5 + p_6 + p_7 + p_8$
6	$p_1 + p_3 + p_4 + p_8$	$p_3 + p_4 + p_5 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_6 + p_8$	$p_2 + p_3 + p_4 + p_5 + p_6 + p_8$	$p_1 + p_4 + p_7 + p_8$	$p_4 + p_5 + p_7 + p_8$	$p_1 + p_2 + p_4 + p_6 + p_7 + p_8$	$p_2 + p_4 + p_5 + p_6 + p_7 + p_8$
7	$p_2 + p_3 + p_4 + p_8$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_8$	$p_3 + p_4 + p_6 + p_8$	$p_1 + p_3 + p_4 + p_5 + p_6 + p_8$	$p_2 + p_4 + p_7 + p_8$	$p_1 + p_2 + p_4 + p_5 + p_7 + p_8$	$p_4 + p_6 + p_7 + p_8$	$p_1 + p_4 + p_5 + p_6 + p_7 + p_8$
8	$p_1 + p_2 + p_3 + p_4 + p_8$	$p_2 + p_3 + p_4 + p_5 + p_8$	$p_1 + p_3 + p_4 + p_6 + p_8$	$p_3 + p_4 + p_5 + p_6 + p_8$	$p_1 + p_2 + p_4 + p_7 + p_8$	$p_2 + p_4 + p_5 + p_7 + p_8$	$p_1 + p_4 + p_6 + p_7 + p_8$	$p_4 + p_5 + p_6 + p_7 + p_8$
9	$p_1 + p_8$	$p_5 + p_8$	$p_1 + p_2 + p_6 + p_8$	$p_2 + p_5 + p_6 + p_8$	$p_1 + p_3 + p_7 + p_8$	$p_3 + p_5 + p_7 + p_8$	$p_1 + p_2 + p_3 + p_6 + p_7 + p_8$	$p_2 + p_3 + p_5 + p_6 + p_7 + p_8$
10	$p_2 + p_8$	$p_1 + p_2 + p_5 + p_8$	$p_6 + p_8$	$p_1 + p_5 + p_6 + p_8$	$p_2 + p_3 + p_7 + p_8$	$p_1 + p_2 + p_3 + p_5 + p_7 + p_8$	$p_3 + p_6 + p_7 + p_8$	$p_1 + p_3 + p_5 + p_6 + p_7 + p_8$
11	$p_1 + p_2 + p_8$	$p_2 + p_5 + p_8$	$p_1 + p_6 + p_8$	$p_5 + p_6 + p_8$	$p_1 + p_2 + p_3 + p_7 + p_8$	$p_2 + p_3 + p_5 + p_7 + p_8$	$p_1 + p_3 + p_6 + p_7 + p_8$	$p_3 + p_5 + p_6 + p_7 + p_8$
12	$p_3 + p_8$	$p_1 + p_3 + p_5 + p_8$	$p_2 + p_3 + p_6 + p_8$	$p_1 + p_2 + p_3 + p_5 + p_6 + p_8$	$p_7 + p_8$	$p_1 + p_5 + p_7 + p_8$	$p_2 + p_6 + p_7 + p_8$	$p_1 + p_2 + p_5 + p_6 + p_7 + p_8$
13	$p_1 + p_3 + p_8$	$p_3 + p_5 + p_8$	$p_1 + p_2 + p_3 + p_6 + p_8$	$p_2 + p_3 + p_5 + p_6 + p_8$	$p_1 + p_7 + p_8$	$p_5 + p_7 + p_8$	$p_1 + p_2 + p_6 + p_7 + p_8$	$p_2 + p_5 + p_6 + p_7 + p_8$
14	$p_2 + p_3 + p_8$	$p_1 + p_2 + p_3 + p_5 + p_8$	$p_3 + p_6 + p_8$	$p_1 + p_3 + p_5 + p_6 + p_8$	$p_2 + p_7 + p_8$	$p_1 + p_2 + p_5 + p_7 + p_8$	$p_6 + p_7 + p_8$	$p_1 + p_5 + p_6 + p_7 + p_8$
15	$p_1 + p_2 + p_3 + p_8$	$p_2 + p_3 + p_5 + p_8$	$p_1 + p_3 + p_6 + p_8$	$p_3 + p_5 + p_6 + p_8$	$p_1 + p_2 + p_7 + p_8$	$p_2 + p_5 + p_7 + p_8$	$p_1 + p_6 + p_7 + p_8$	$p_5 + p_6 + p_7 + p_8$

Theorem 3 (Compression Theorem)**.** The $K \times N$ matrix **PubMat** can be represented in the following compressed format:

1. A list of 2ν elements of the vector $\mathbf{p}_{base} = (cg_0, cg_1, \dots, cg_{\nu-1}, dg_0, dg_1, \dots, dg_{\nu-1})$
2. A list of K coset leaders $\mathbf{c}_0, \dots, \mathbf{c}_{K-1} \in \{0,1\}^\nu$,
3. Any element $p_{i,j}$ is represented by its coset leader $\mathbf{c}_i$ and another ν-dim binary vector $\mathbf{d}_i$ such that $p_{i,j} = \mathbf{p}_{base} \cdot (\mathbf{c}_i + \mathbf{d}_i, \mathbf{d}_i)$.

Proof. Without a loss of generality, we can proof the theorem by analyzing the first two elements of the first row of the matrix **PubMat**, $p_{0,0}$ and $p_{0,1}$. Let us denote the first row of the matrix $\mathbf{G}$ as $\mathbf{g}_0$, and the first and second column of the matrix $\mathbf{P}_{c,d}$ with $\mathbf{Col}_0$ and $\mathbf{Col}_1$. Then we can represent the elements $p_{0,0}$ and $p_{0,1}$ as the following dot products:

$$p_{0,0} = \mathbf{g}_0 \cdot \mathbf{Col}_0, \text{ and, } p_{0,1} = \mathbf{g}_0 \cdot \mathbf{Col}_1.$$

Since the elements of the vector $\mathbf{g}_0$ are from the binary linear span set $\mathcal{G}_{priv} = \{g \mid g = \mathbf{g}_{base} \cdot \mathbf{b},\ \ \mathbf{b} \in \{0,1\}^\nu\}$, we can represent the vector $\mathbf{g}_0$ as follows:

$$\mathbf{g}_0 = \mathbf{g}_{base} \cdot \mathbf{B}_0$$

where $\mathbf{B}_0$ is an $\nu \times N$ binary matrix (every column in the matrix $\mathbf{B}_0$ represent one ν-dim binary vector $\mathbf{b}$ used in the definition of the binary linear span $\mathcal{G}_{priv}$).

Since we control the generation of the matrix $\mathbf{B}_0$ we can assume that its rank is ν.

Next, let us denote by $(\mathbf{c}_0, \mathbf{d}_0)$ and $(\mathbf{c}_1, \mathbf{d}_1)$ the 2ν-dim vectors that multiplies the vector $\mathbf{p}_{base}$ and gives $p_{0,0}$, and $p_{0,0}$ i.e.,

$$p_{0,0} = \mathbf{p}_{base} \cdot (\mathbf{c}_0, \mathbf{d}_0), \text{ and, } p_{0,1} = \mathbf{p}_{base} \cdot (\mathbf{c}_1, \mathbf{d}_1).$$

Let us partition all $2^{2\nu}$ elements of the binary linear span of $\mathbf{p}_{base}$ as in Theorem 2 and let us choose the element $(\mathbf{c}_0, \mathbf{d}_0)$ as a coset leader. Then, from the fact that the rank of the matrix $\mathbf{B}_0$ is ν, it follows that $\mathbf{c}_0$, $\mathbf{c}_1$ and $\mathbf{d}_1$ as three ν-dim vectors must be linearly dependent. More concretely it follows that

$$\mathbf{c}_1 = \mathbf{c}_0 + \mathbf{d}_1.$$

The same reasoning can be applied for every other element $p_{0,j}$ of the first row, i.e., by keeping the fixed element $(\mathbf{c}_0, \mathbf{d}_0)$ as coset leader, for the corresponding 2ν-dim vector $(\mathbf{c}_j, \mathbf{d}_j)$ we have that

$$\mathbf{c}_j = \mathbf{c}_0 + \mathbf{d}_j.$$

So, by keeping the element $\mathbf{c}_0$, and then having a list of N elements $\mathbf{d}_0, \dots, \mathbf{d}_{N-1}$ we can represent the first row of the matrix **PubMat** in a compressed format. The same reasoning applies to every row of **PubMat**. ☐

The procedure described in the proof of the previous Theorem 3 for obtaining the compressed format is denoted by `CompressPublic`(**PubMat**), and the inverse procedure that obtains **PubMat** from the compressed format is denoted by `DecompressPublic`(**PubMat**).

Design decision: When generating **PubMat**, if $K \leq 2^{\nu}$, then we randomly choose its rows such that the list of K coset leaders $\mathbf{c}_0, \ldots, \mathbf{c}_{K-1}$ are all different. On the other hand, if $K \geq 2^{\nu}$, then we randomly choose its rows such that the first K coset leaders are all different, then the next K coset leaders are all different, and so on until the last subset of coset leaders (that might not has K elements) are all different.

4.4 Definition of encoding procedure for achieving the goal (G4.)

In a typical McEliece public key scheme with a binary matrix of size $K \times N$, the binary error vector $\mathbf{error} = (e_0, e_1, \ldots, e_{N-1})$ has a small Hamming weight t in comparison to the length of the code N. That enables to define the *Information Set Decoding* attack: Find an information set of indexes $\mathcal{I} = \{i_0, i_1, \ldots, i_{K-1}\}$, such that the projection of the error vector over that information set $\mathbf{error}_{\mathcal{I}} = \mathbf{0}$. That means the encoded message can be simply decoded as:

$$\mathbf{m} = \mathbf{c}_{\mathcal{I}} \cdot (\mathbf{PubMat}_{\mathcal{I}})^{-1}$$

i.e. by multiplying the projection of the ciphertext $\mathbf{c}$ over the information set $\mathcal{I}$ with the inverse matrix of the projection of **PubMat** over the information set $\mathcal{I}$. In order to make the probability of guessing a correct information set $\mathcal{I}$, less than 2^{-128} or less than 2^{-192}, some big values of N and K would be necessary, thus making the public key big.

Our approach for addressing the Information Set Decoding attack is the following: Construct the error vector **error** such that the all bits in the final ciphertext can be affected (or abandon the Hamming weight metrics for the error vector). We achieve that approach as follows.

Definition 10. Let $L \geq 2$ be an even number. Define a L-dim vector

$$\mathbf{e}_{base} = (e_0, e_1, \ldots, e_{L-1})$$

by first choosing the first two elements

$$(e_0, e_1) \xleftarrow{\$} (\mathbb{F}_q)^2$$

from $\mathbb{F}_q$ uniformly at random. Then, every subsequent pair define as

$$(e_{2i}, e_{2i+1}) \leftarrow SHA2(e_{2i-2} || e_{2i-1}) \text{ for } i \in \{1, \ldots, \frac{L}{2} - 1\}.$$

Let the set of the elements in $\mathbf{e}_{base}$ be $\mathcal{E}_{base} = \{e_0, e_1, \ldots, e_{L-1}\}$. A binary linear span of $\mathcal{E}_{base}$ is the set

$$\mathcal{E}_{BinSpan} = \{e \mid e = \mathbf{e}_{base} \cdot \mathbf{b}, \ \mathbf{b} \in \{0,1\}^L\},$$

where the operation $\cdot$ is a dot product.

Definition 11. The error vector **error** $\in (\mathbb{F}_q)^N$ is defined as

$$\mathbf{error} \xleftarrow{\$} (\mathcal{E}_{base})^N.$$

The following corollary follows directly from the Definition 11:

Corollary 2. *The error vector* **error** *can be represented as a product of the vector* $\mathbf{e}_{base}$ *and a* $L \times N$ *binary matrix* **B** *chosen uniformly at random* $\mathbf{B}_{L\times N} \xleftarrow{\$} \{0,1\}^{L\times N}$*, i.e.,*

$$\mathbf{error} = \mathbf{e}_{base} \cdot \mathbf{B}.$$

Design decision: By keeping the elements of the set $\mathcal{E}_{base}$ unknown, an Information Set Decoding attack is rendered as inapplicable.

While in the process of creating the public and private key, the creator of the key-pair can control the number of elements a and b in every row of the matrix $\mathbf{P}_{a,b}$ to be odd, and the Hamming weight of every column in the binary matrix **H** to be odd, the creator does not control the encryption party that can construct the binary matrix **B** such that its rows can have either odd or even Hamming weight. Thus it is easy to get the following

Corollary 3. *All elements of the* $L \times R$ *matrix* $\mathbf{R} = \mathbf{B} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H}$ *form the following set:* $\{0, a, b, a+b\} \subset \mathbb{F}_q$.

Definition 12. Set $\mathbf{e}' = \mathbf{C} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} = (e'_1, \ldots, e'_R)$, and define $Candidates'$ be the following binary linear span set of the elements in $\mathbf{e}'$: $Candidates' = \{s \mid \exists v \in \{0,1\}^R,\ \ s = v \cdot \mathbf{e}'\}$.

Proposition 7. *If the matrices* $\mathbf{B}_{L\times N}$ *from the Corollary 2 and* $\mathbf{R}_{L\times R}$ *from the Corollary 3 have a full rank* L*, then the elements* $(a+b)e_i$*, for* $i = 0, 1, \ldots, L-1$ *belong to the linear binary span* $Candidates'$.

Proof. We start with

$$\mathbf{e}' = \mathbf{C} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = (\mathbf{m} \cdot \mathbf{PubMat} + \mathbf{error}) \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{m} \cdot \mathbf{PubMat} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} + \mathbf{error} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{m} \cdot \mathbf{G} \cdot \mathbf{P}_{c,d} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} + \mathbf{error} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{m} \cdot \mathbf{G} \cdot \mathbf{I} \cdot \mathbf{H} + \mathbf{error} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{m} \cdot \mathbf{G} \cdot \mathbf{H} + \mathbf{error} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{m} \cdot \mathbf{0} + \mathbf{error} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{error} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$

$$\mathbf{e}' = \mathbf{e}_{base} \cdot \mathbf{B} \cdot \mathbf{P}_{a,b} \cdot \mathbf{H} \Rightarrow$$
$$\mathbf{e}' = \mathbf{e}_{base} \cdot \mathbf{R}.$$

Under the assumption that the matrices **B** and **R** have full rank L it follows that the elements $(a+b)e_i$, for $i = 0, 1, \ldots, L-1$ can be obtained as a binary linear combination of elements in the vector $\mathbf{e}'$. ☐

As an immediate consequence we have the following

Corollary 4. Let the set *Candidates* be defined as $Candidates = \{\frac{s}{a+b} \mid s \in Candidates'\}$. If the matrices $\mathbf{B}_{L\times N}$ from the Corollary 2 and $\mathbf{R}_{L\times R}$ from the Corollary 3 have a full rank L, then the elements e_i, for $i = 0, 1, \ldots, L-1$ belong to the set *Candidates*.

Considering the full ranks of the matrices **B** and **R**, without a proof we give the following proposition

Proposition 8. If the matrix **B** does not have a full rank L, then the matrix **R** does not have a full rank L. ☐

A natural question comes in connection with the error vector: If we have all these definitions, how do we encode some information that can be a shared information between the encoder and decoder? To answer that question we use the following definition:

As a consequence of all previous definitions, corollaries and propositions in this section we have the following theorem

Theorem 4 (Probability of the unsuccessful decoding)**.** Let the ciphertext pair **Ciphertext** $= (\mathbf{C}, h)$ be produced as described in the algorithm **Encapsulate** EDON-$\mathcal{K}$ in Table 2. The probability for unsuccessful decoding is bounded by the following expression:

$$\prod_{i=N-L+1}^{N} \left(1 - \frac{1}{2^i}\right) \le P_{unsucc}(\text{EDON-}\mathcal{K}) < \prod_{i=R-L+1}^{R} \left(1 - \frac{1}{4^i}\right) \tag{5}$$

Proof. To prove the theorem we use the results of [3] for the rank of random matrices over finite fields. The left part in the inequality expression (5)

$$\prod_{i=N-L+1}^{N} \left(1 - \frac{1}{2^i}\right)$$

is addressing the probability the binary matrix $\mathbf{B}_{L\times N}$ to not have a full rank L. On the other hand, from Corollary 3 we have that the elements in **R** are $\{0, a, b, a+b\}$, which form a group of 4 elements that is isomorphic with the additive group of $GF(4)$. Thus, as a first and not so tight bound we can take the probability that a random matrix of size $L \times R$ in $GF(4)$ do not have a full rank L:

$$\prod_{i=R-L+1}^{R} \left(1 - \frac{1}{4^i}\right).$$

□

Definition 13. We define a function of upper bound of failure probability to be a function of three variables L, R and Q as follows:

$$\mathcal{U}(L, R, Q) \equiv \prod_{i=R-L+1}^{R} \left(1 - \frac{1}{Q^i}\right). \tag{6}$$

Design decision: While the numerical experiments show that the failure probability for some concrete values of L and R can be modeled with the function $\mathcal{U}(L, R, Q)$ for some value Q in the interval $[8.0, 16.0]$, for our submission of EDON-$\mathcal{K}$ we took a conservative value for $Q = 4$, and decided to choose parameters for L and R such that the probability of unsuccessful decoding is upper-bounded by 2^{-64}.

More concretely, for $\mathbb{F}_q = GF(2^{128})$, we choose $L = 6$, $R = 40$, and

$$\mathcal{U}(6, 40, 4) < 2^{-69}.$$

For $\mathbb{F}_q = GF(2^{192})$, we choose $L = 8$, $R = 40$, and

$$\mathcal{U}(8, 40, 4) < 2^{-65}.$$

We want to emphasize that for the chosen L and R, the empirical experiments indicate the failure probability is significantly lower than 2^{-64}, but a rigorous proof for those bounds is an open problem. We choose to claim at least a 2^{-64} failure probability in connection with the part 4.A.2 Security Definition for Encryption/Key-Establishment, of the NIST call: *For the purpose of estimating security strengths, it may be assumed that the attacker has access to the decryptions of no more than* 2^{64} *chosen ciphertexts.* Thus, with our choice of parameters L and R, the probability of decoding failure of EDON-$\mathcal{K}$ implies that the expected number of decoding attempts before a failure occurs will be much bigger than 2^{64}.

For a value $Q = 12$ (as a middle value between 8 and 16) we give the following estimations for the decoding failure probability:

For $\mathbb{F}_q = GF(2^{128})$, $L = 6$, $R = 40$,

$$\mathcal{U}(6, 40, 12) < 2^{-125}. \tag{7}$$

For $\mathbb{F}_q = GF(2^{192})$, $L = 8$, $R = 40$,

$$\mathcal{U}(8, 40, 12) < 2^{-118}. \tag{8}$$

During the standardization process, if there is a need to decrease the failure probability further, the internal parameter R can be tweaked to the values $R = 48$ or $R = 56$ or even $R = 64$, without affecting the size of the public keys (but there will be small consequences on key generation time and on decoding time).

4.5 EDON-$\mathcal{K}$ is IND-CCA2 secure by design (goal G5.)

The designing process for EDON-$\mathcal{K}$ was guided by the goal the scheme to be non-malleable. The notion of non-malleability was proposed by Dolev, Dwork and Naor in [5] and then, in [1] Bellare et al., proved the equivalence between non-malleability (NM-CCA2) and indistinguishability under adaptive chosen ciphertext attack (IND-CCA2).

A public-key cryptosystem is non-malleable if given a ciphertext, it is no easier to generate a different ciphertext so that the respective plaintexts are related, than it is to do so without access to the ciphertext.

Let us recall the format of the ciphertext in EDON-$\mathcal{K}$: **Ciphertext** $= (\mathbf{C}, h)$, where $\mathbf{C} = \mathbf{M} \cdot \mathbf{PubMat} + \mathbf{error}$, and $h = SHA2(s_1||s_0||SHA2(\mathbf{C}))$.

By setting $h = SHA2(s_1||s_0||SHA2(\mathbf{C}))$ we protect h and $\mathbf{C}$ of being malleable, and this non-malleability feature depends on the strength of the cryptographic function $SHA2$.

Let us recall the essence of the IND-CCA2 game in the KEM context: Upon receiving an information (a sequence of bits) from the challenger, the adversary's job is to decide whether the information was a valid ciphertext and its corresponding encapsulated key, or the information was just a bunch of randomly generated bits. The adversary can send queries for decryption to the challenger before and after receiving that information.

Having in mind that in EDON-$\mathcal{K}$, $SharedSecret = SHA2(s_0||s_1||SHA2(\mathbf{C}))$ and **Ciphertext** $= (\mathbf{C}, h)$, upon receiving the triplet $(\mathbf{C}, h, SharedSecret)$, to successfully distinguish it from a randomly generated sequence of bits (with the same length), means that the adversary has an efficient polynomial algorithms for solving at least one of these two problems:

1. For a randomly chosen **M** and randomly chosen **error** defined by Encapsulate algorithm, and a randomly chosen **PubMat** defined by KeyGen algorithm, distinguish $\mathbf{C} = \mathbf{M} \cdot \mathbf{PubMat} + \mathbf{error}$ from a random sequence of bits (with the same length);
2. Find $SHA2$ preimages for $SharedSecret$ in a format: $s_0||s_1||SHA2(\mathbf{C})$, and check if $h = SHA2(s_1||s_0||SHA2(\mathbf{C}))$.

It seems that neither problem has an efficient polynomial solution. A formal proof is being prepared and will appear in a paper.

5 Detailed performance analysis (2.B.2)

5.1 Description of platform

The measurements were collected using `supercop-20170904` running on a computer with a CPU Intel® Core™ i7-7600U CPU at 2.90 GHz, with Turbo Boost disabled. The machine has 32GB of RAM and runs Ubuntu 17.04. Benchmarks used `crypto_kem`, which ran on one core of the CPU.

The compiler option was just the first one:
`gcc -march=native -mtune=native -O3 -fomit-frame-pointer -fwrapv`

`LOOPS` in `crypto_kem/try.c` was reduced to 0 for `SMALL` and 1 otherwise, and `TIMINGS` in `crypto_kem/measure.c` was reduced to 1.

5.2 Space and Time

In Table 9 and Table 10 we give the size of the public key, the private key and the ciphertext, as well as the number of cycles for KeyGen, Encapsulate and Decapsulate for the reference submissions of Edon-$\mathcal{K}$128 and Edon-$\mathcal{K}$192. We also give those numbers for several alternative choices of the parameters.

	Edon-K128 space			Keygen (cycles) K:thousands M:millions	Encapsulate (cycles) K:thousands M:millions	Decapsulate (cycles) K:thousands M:millions
Submission folder name	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext Size (bytes)			
edonk128ref	**2576**	**32**	**2336**	**2.5 M**	**576 K**	**28.7 M**
edonk128K16N80nu8L6	1552	32	1312	1.6 M	1.3 M	23.4 M
edonk128K08N72nu8L8	840	32	1184	950 K	143 K	634 M
edonk128K32N96nu4L4	1680	32	1568	1.9 M	372 K	2.0 M
edonk128K16N80nu4L6	776	32	1312	1.0 M	158 K	35.0 M

Table 9: Size of the keys, size of the ciphertext, and number of cycles for KeyGen, Encapsulate and Decapsulate for Edon-$\mathcal{K}$128.

	Edon-K192 space			Keygen (cycles) K:thousands M:millions	Encapsulate (cycles) K:thousands M:millions	Decapsulate (cycles) K:thousands M:millions
Submission folder name	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext Size (bytes)			
edonk192ref	**2192**	**32**	**2736**	**2.0 M**	**496 K**	**54.6 M**
edonk192K48N144nu4L4	3672	32	3504	4.6 M	917 K	3.9 M
edonk192K32N128nu4L6	2256	32	3120	2.1 M	566 K	44.0 M
edonk192K16N112nu4L8	1096	32	2736	1.4 M	303 K	268 M

Table 10: Size of the keys, size of the ciphertext, and number of cycles for KeyGen, Encapsulate and Decapsulate for EDON-$\mathcal{K}$192.

5.3 How parameters affect performance

It is obvious that the size of the public key is mostly affected by the choice of the parameter ν - which determines the size the set $\mathcal{P}_{base} = \{cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}\}$, and which further affects the level of the compression of the public key. For $\nu = 4$ the public key size is approximately half of the size of the public key when $\nu = 8$.

The parameter L affects the speed of Decapsulate procedure. The lower the parameter L is, the faster the Decapsulate procedure is. However, too low L would significantly affect the security of the scheme, due to the complexity of the direct Gaussian elimination attack given in Corollary 6 which is $O(2^{L(K+L)})$.

5.4 Optimizations

The submission of the reference C code is lightly optimized, and the same code is given in the Optimal Implementation folder. The speeds measured and presented in Table 9 and Table 10 are decent and comparable with other post-quantum KEM schemes. The implementations that include AVX operations, will be given after the November 30 2017 deadline [12].

We expect those optimizations to speed-up the scheme 2 to 6 times.

6 Expected strength (2.B.4) for each parameter set

In Table 11 we give the expected strength for our reference proposal for EDON-$\mathcal{K}$128. We also give the expected strengths for several alternative choices of parameters. All of these parameters sets have expected strength that belong at least in Category 1 of the NIST call [12].

	Edon-K128 parameter space and security claims									Attack: Finding an equivalent PrivMat matrix		Parity Check Matrix Attack, log2	Direct Gausian Elimination attack, log2
Submission folder name	K	N	v	q GF(2^q)	R	L	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext Size (bytes)	Struture of PrivMat columns via Finite Field Vector Subset Ratio Problem, log2	Guess of constants a, b, c and d, log2		
	32	144	8	128	40	4	4896	32	2336	144	128	224	144
	32	96	8	128	40	4	3360	32	1568	96	128	128	144
	16	144	8	128	40	8	2576	32	2336	144	128	256	192
edonk128ref	**16**	**144**	**8**	**128**	**40**	**6**	**2576**	**32**	**2336**	**144**	**128**	**256**	**132**
edonk128K16N80nu8L6	16	80	8	128	40	6	1552	32	1312	80	128	128	132
	8	128	8	128	40	8	1288	32	2080	128	128	240	128
edonk128K08N72nu8L8	8	72	8	128	40	8	840	32	1184	72	128	128	128
	32	144	4	128	40	4	2448	32	2336	144	128	224	144
	32	128	4	128	40	4	2192	32	2080	128	128	192	144
edonk128K32N96nu4L4	32	96	4	128	40	4	1680	32	1568	96	128	128	144
	16	128	4	128	40	6	1160	32	2080	128	128	224	132
	16	96	4	128	40	6	904	32	1568	96	128	160	132
edonk128K16N80nu4L6	16	80	4	128	40	6	776	32	1312	80	128	128	132
	8	80	4	128	40	8	452	32	1312	80	128	144	128
	8	72	4	128	40	8	420	32	1184	72	128	128	128

Table 11: The parameter space for EDON-$\mathcal{K}$128.

Also we give a Table 12 for the corresponding reference proposal for EDON-$\mathcal{K}$192, and severak alternative choices of parameters. In that table, all parameters sets have expected strength that belong at least in Category 3 of the NIST call [12].

	Edon-K192 parameter space and security claims									Attack: Finding an equivalent PrivMat matrix		Parity Check Matrix Attack, log2	Direct Gausian Elimination attack, log2
Submission folder name	K	N	v	q GF(2^q)	R	L	Public Key Size (bytes)	Private Key Size (bytes)	Ciphertext Size (bytes)	Struture of PrivMat columns via Finite Field Vector Subset Ratio Problem, log2	Guess of constants a, b, c and d, log2		
	44	144	8	192	40	4	6764	32	3504	144	192	200	192
	32	144	8	192	40	6	5024	32	3504	144	192	224	228
	28	128	8	192	40	6	3996	32	3120	128	192	200	204
edonk192ref	**16**	**112**	**8**	**192**	**40**	**8**	**2192**	**32**	**2736**	**112**	**192**	**192**	**192**
	16	144	8	192	40	8	2704	32	3504	144	192	256	192
edonk192K48N144nu4L4	48	144	4	192	40	4	3672	32	3504	144	192	192	208
edonk192K32N128nu4L6	32	128	4	192	40	6	2256	32	3120	128	192	192	228
	28	144	4	192	40	6	2222	32	3504	144	192	232	204
	24	144	4	192	40	8	1932	32	3504	144	192	240	256
	20	128	4	192	40	8	1482	32	3120	128	192	216	224
	16	128	4	192	40	8	1224	32	3120	128	192	224	192
edonk192K16N112nu4L8	16	112	4	192	40	8	1096	32	2736	112	192	192	192

Table 12: The parameter space for EDON-$\mathcal{K}$192.

7 Analysis of known attacks (2.B.5)

7.1 Information set decoding attack is not applicable on EDON-$\mathcal{K}$

This part has been addressed in Section 4.4.

7.2 Attacks on the private key structure

The attack for finding an equivalent private matrix **PrivMat** consists of two parts:

1. Revealing the binary structure of **PrivMat** or for some equivalent annihilator matrix $\mathbf{PrivMat}'$ to the public key matrix **PubMat**. By "the binary structure" we mean reviling which positions in the matrix **PrivMat** are occupied by the constant a, and which positions are occupied by the constant b.
2. Finding the constants a and b.

7.2.1 Revealing the binary structure of PrivMat

Let us recall that in EDON-$\mathcal{K}$ the ciphertext vector $\mathbf{C}$ has to be multiplied by the matrix $\mathbf{PrivMat} = \mathbf{P}_{a,b} \cdot \mathbf{H}$. As we already discussed in 4.4, the number of elements a and b in every row of the matrix $\mathbf{P}_{a,b}$ is odd, and the Hamming weight of every column in the binary matrix $\mathbf{H}$ is also odd. That means the entries of the matrix **PrivMat** are either a or b.

Additionally, let us recall Corollary 1: the matrix **PrivMat** is annihilator for the public key matrix **PubMat** i.e.

$$\mathbf{0} = \mathbf{PubMat} \cdot \mathbf{PrivMat}.$$

Let us also recall Theorem 3 and the fact that the public key in the compressed form has a 2ν-dim vector $\mathbf{p}_{base} = (cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}) = (p_1, \ldots, p_\nu, p_{\nu+1}, \ldots, p_{2\nu})$. From this information the ratio $\frac{c}{d}$ can be obtained as:

$$\frac{c}{d} = \frac{p_1}{p_{\nu+1}}.$$

From equation (1) we have that

$$\frac{c}{d} = \frac{a}{b}.$$

Proposition 9. Let $p_i,\ i \in \{0, 1, \ldots, K-1\}$ are the rows of the matrix **PubMat**, let $q_j,\ j \in \{0, 1, \ldots, N-1\}$ are the columns of the matrix **PrivMat**, and let for every

$j \in \{0, 1, \ldots, N-1\}$, the sets $S_1^j, S_2^j \subset \mathcal{I} = \{0, 1, \ldots, N-1\}$ are defined as:

$$\begin{cases} S_1^j = & \{i \mid q_{i,j} = a\} \\ S_2^j = & \{i \mid q_{i,j} = b\}. \end{cases} \tag{9}$$

Then

$$a \sum_{j \in S_1^j} p_{i,j} + b \sum_{j \in S_2^j} p_{i,j} = 0 \tag{10}$$

Knowing this, an attacker that has the public matrix **PubMat** can try to solve the following problem:

Definition 14. (FINITE FIELD VECTOR SUBSET RATIO PROBLEM (FFVSRP)) Let **PubMat** $= [p_{i,j}]_{K \times N}$ be the public $K \times N$ matrix in non-compressed form, and let $\mathcal{I} = \{0, 1, \ldots, N-1\}$ be an index set. Find a two-way split of the index set $\mathcal{I}$ i.e., find two disjunctive subsets S_1 and S_2 ($S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = \mathcal{I}$) such that

$$\frac{\sum_{j \in S_1} p_{i,j}}{\sum_{j \in S_2} p_{i,j}} = \frac{d}{c}, \text{ for every row } i \in \{0, 1, \ldots, K-1\}. \tag{11}$$

By solving the FFVSRP attacker reveals a part of the structure of the private matrix **PubMat**. More precisely, it reveals the structure of the columns of the **PubMat** (or some equivalent **PubMat**$'$), on which position there is a constant a and on which position there is a constant b. Note that the attacker still do not have the concrete values for a and b that are also crucial for a successful decoding procedure.

It is easy to see that a SUBSET-SUMS RATIO PROBLEM is a special case of FINITE FIELD VECTOR SUBSET RATIO PROBLEM (just put $a = 0$ and $b = 1$).

Thus FINITE FIELD VECTOR SUBSET RATIO PROBLEM is a *NP-hard problem*.

Considering the possible heuristics for solving FFVSRP, it seems that there is no known heuristics, except the exhaustive search in all 2^N subset pairs (S_1, S_2), and checking if the relation (11) holds.

In the literature there is a related SUBSET-SUMS RATIO PROBLEM, defined by Woeginger and Yu [15], but FFVSRP is different in at least three aspects: 1. it asks simultaneously to satisfy splitting of K sets (not just one set); 2. it asks the ratios between the sums to be exact (not approximate); 3. it is defined over finite fields.

It also seems that the heuristics of splitting the zero sums, that Bernstein et al., applied for quantum attacks on the SUBSET-SUMS PROBLEM in [2] is not applicable for the FINITE FIELD VECTOR SUBSET RATIO PROBLEM.

7.2.2 Finding the constants a and b

The constants a, b, c and d are connected with the relation $\frac{c}{d} = \frac{a}{b}$. From the public information about 2ν-dim vector $\mathbf{p}_{base} = (cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}) = (p_1, \ldots, p_\nu, p_{\nu+1}, \ldots, p_{2\nu})$ there is not enough information to find the values for c and d. Actually, the following property holds:

Proposition 10. For a given $\mathbf{p}_{base} = (cg_0, cg_1, \ldots, cg_{\nu-1}, dg_0, dg_1, \ldots, dg_{\nu-1}) = (p_1, \ldots, p_\nu, p_{\nu+1}, \ldots, p_{2\nu}) \in (\mathbb{F}_q)^{2\nu}$, there are $q-1$ solutions for the variables c, d, $g_0, \ldots, g_{\nu-1}$.

Proof. Just put some concrete value for the constant c. Then, by the relations from $\mathbf{p}_{base}$ the solution for d and $g_0, \ldots, g_{\nu-1}$ follows. ☐

From this discussion we conclude that the attack for finding an equivalent private matrix **PrivMat** has a cost that is comparable with the cost of finding a 128 bit of AES-128 or finding a 192 bit key of AES-192 (Category 1 and Category 3 in the NIST call [12]).

7.3 Attacks on the ciphertext

7.3.1 Attacks using the parity check matrix of the public key matrix PubMat

The decoding i.e. the Decapsulate procedure uses the matrix **PrivMat** as an annihilator matrix for **PubMat** and the knowledge of the values of the constants a and b in order to compute the *SharedSecret*.

Since **PubMat** is a generator matrix, the attacker can use the transpose of its parity check matrix $\mathbf{Parity}^T$ as its annihilator matrix. More concretely, for an $K \times N$ generator matrix **PubMat**, its parity check matrix **Parity** is $(N-K) \times N$ matrix such that

$$\mathbf{0} = \mathbf{PubMat} \cdot \mathbf{Parity}^T, \tag{12}$$

where $\mathbf{0}$ is $K \times (N-K)$ zero matrix.

The attack is given as the algorithm described in Table 13.

Corollary 5. The complexity of the Parity Check Matrix Attack is $O(2^{2(N-K)})$.

Proof. From the Step 2 we have that the number of elements in the set *Candidates* is $2^{(N-K)}$. Since in Step 3 we need to find a pair of elements from *Candidates* that after up to $\frac{L}{2}$ consecutive hash computations will give the expected hash value h, it follows that the expected number of trials is $O(2^{2(N-K)})$. ☐

```
Parity Check Matrix Attack
Input: Ciphertext = $(\mathbf{C}, h)$,
Output: $SharedSecret$.

1. Compute $\mathbf{e} = \mathbf{C} \cdot \mathbf{Parity}^T = (e_1, \dots, e_{N-K})$
2. Set $Candidates = \{s \mid \exists \mathbf{b} \in \{0,1\}^{N-K},\ s = \mathbf{b} \cdot \mathbf{e}\}$
3. Find a pair $(s_\mu, s_\nu) \in Candidates \times Candidates$ such that there exist $1 \leq i \leq \frac{L}{2}$ such that $(s_0, s_1) = \underbrace{SHA2(\dots SHA2(s_\mu, s_\nu))}_{i}$ and $h = SHA2(s_1||s_0||SHA2(\mathbf{C}))$
    3.1 Set $SharedSecret = SHA2(s_0||s_1||SHA2(\mathbf{C}))$
    3.2 Return $SharedSecret$.
```

Table 13: The parity check matrix attack

7.3.2 Attacks by direct Gaussian elimination

Definition 15. Let **PubMatShort** be the following $(K+L) \times (K+L)$ matrix:

$$\mathbf{PubMatShort} = \begin{bmatrix} \mathbf{PubMat}_{[0,\dots,(K+L-1)]} \\ \mathbf{B}_{[0,\dots,(K+L-1)]} \end{bmatrix}, \tag{13}$$

where $\mathbf{PubMat}_{[0,\dots,(K+L-1)]}$ represents the first $K+L$ columns of the matrix **PubMat**, and $\mathbf{B}_{[0,\dots,(K+L-1)]}$ is some $L \times (K+L)$ binary matrix. Let the vector $\mathbf{C}_{[0,\dots,(K+L-1)]}$ represent the first $K+L$ components of the vector **C**.

The attacker knows the value of $\mathbf{PubMat}_{[0,\dots,(K+L-1)]}$, but does not know the value of $\mathbf{B}_{[0,\dots,(K+L-1)]}$ which is actually the first part of the binary $L \times N$ matrix **B** used in the encapsulation process.

The attack goes like this: Guess the value of the matrix $\mathbf{B}_{[0,\dots,(K+L-1)]}$, set an unknown vector $\mathbf{X} = (m'_0, m'_1, \dots, m'_{K-1}, e'_0, e'_1, \dots, e'_{L-1})$ then solve the following linear system:

$$\mathbf{X} \cdot \mathbf{PubMatShort} = \mathbf{C}_{[0,\dots,(K+L-1)]}. \tag{14}$$

From (e'_{L-2}, e'_{L-1}) compute the $SharedSecret' = (s_0, s_1) = SHA2(e'_{L-2}||e'_{L-1})$, and check if $h' = \overline{SHA2}(s_0, s_1) == h$. Repeat the search with new randomly guessed matrices $\mathbf{B}_{[0,\dots,(K+L-1)]}$ until you find the $SharedSecret$.

Corollary 6. Assuming that the complexity of inverting an $(K+L) \times (K+L)$ matrix is $O(1)$, the direct Gaussian elimination attack on EDON-$\mathcal{K}$ has a complexity $O(2^{L(K+L)})$.

Proof. It is a simple observation that the size of the unknown binary matrix $\mathbf{B}_{[0,\dots,(K+L-1)]}$ that is guessed in every attempt in the attack, is $L \times (K+L)$. ☐

8 Advantages and limitations (2.B.6)

8.1 Advantages

Small key sizes and small ciphertext. EDON-$\mathcal{K}$ has relatively small key sizes and small ciphertext sizes in the range from 420 Bytes up to 6 KBytes.

CCA2 by design. EDON-$\mathcal{K}$ is designed to offer CCA2 security without a need of some extra (potentially expensive) CPA-to-CCA transformation.

Drop-in-replacement for classical ephemeral KEM schemes. EDON-$\mathcal{K}$ can easily replace the classical Diffie-Hellman key encapsulation schemes.

Forward secrecy. Since the key production is relatively fast, and the key sizes are relatively small, it can be used for ephemeral key exchanges, thus offering a forward secrecy.

Short period cacheing of ephemeral keys Since EDON-$\mathcal{K}$ is CCA2 secure, a short period cashing of its ephemeral keys is possible, without any devastating security consequences for the scheme.

EDON-$\mathcal{K}$-Stealth By additional price of 128 or 192 bits in the public key, EDON-$\mathcal{K}$ can offer the following "stealth mode" of operation: **One private key, and millions of corresponding indistinguishable public keys.** The possibility of this mode is just mentioned here, but the source code of its implementation will follow in the upcoming period (depending on the interest of the community for this mode.) The key generation described in Table 1 is changed only in Step 2, Step 3 and Step 13 as follows:

Step 2:

$$\begin{cases} PrivateKey \xleftarrow{\$} \{0,1\}^{256},\ StealthSalt \xleftarrow{\$} \mathbb{F}_q, & \text{if } PrivateKey \text{ does not exist} \\ StealthSalt \xleftarrow{\$} \mathbb{F}_q, & \text{otherwise.} \end{cases}$$

Step 3: $\textsc{InitializeSeedExpander}(HMAC(PrivateKey, StealthSalt))$

Step 13: Set $PublicKey = [\texttt{CompressPublic}(\mathbf{PubMat}), StealthSalt]$

Parallel decoding. Decoding i.e. Decapsulate procedure in EDON-$\mathcal{K}$ computes repeatedly a hash function and tests for the final hash value. These computations can be trivially done in parallel, either in specialized hardware or in existing AVX instructions for Intel processors platforms. While in this submission, there is still no AVX implementation, we are preparing such an implementation and will post it publicly in the forthcoming period.

Reliance on short input fast hash functions EDON-$\mathcal{K}$ heavily uses computations with cryptographic hash function SHA2, but it always feeds the input of SHA2 with a short input string of bits. It will benefit a lot, if there is a standardized strong and ultra-fast cryptographic hash function with short inputs. Some efforts in this directions have been already made by the cryptographic community. More concretely, "Haraka v2", [9] is one proposal for a hash function, by which EDON-$\mathcal{K}$ would benefit a lot in the decoding phase (we anticipate speedups by a factor 10).

8.2 Limitations

New design. EDON-$\mathcal{K}$ is a new design that introduces several new design novelties such as reliance on the FINITE FIELD VECTOR SUBSET RATIO PROBLEM and choosing the error vectors with components from a binary linear span of a small base set of randomly generated values. We hope that in the forthcoming period and during the NIST post-quantum standardization process, the cryptographic community will put a good scrutiny on the security of this new design.

No side-channel analysis yet. As a new design, EDON-$\mathcal{K}$ is still lacking a side-channel analysis. That analysis, once done by experts in that field, would offer measures and proposals for protecting EDON-$\mathcal{K}$ against side-channel attacks.

9 Acknowledgements

I would like to thank Kristian Gjøsteen for his excellent suggestions that improved the security of the design. I would like to thank Katina Kralevska for useful comments that improved the quality of this submission. I would also like to thank Dan Bernstein for publishing the `pqskeleton` package, and allowing me to use the code of `gmp_import.c` and `gmp_export.c` in the C code implementation for EDON-$\mathcal{K}$.

References

[1] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. *Relations among notions of security for public-key encryption schemes*, pages 26–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[2] Daniel J Bernstein, Stacey Jeffery, Tanja Lange, and Alexander Meurer. Quantum algorithms for the subset-sum problem. In *International Workshop on Post-Quantum Cryptography*, pages 16–33. Springer, 2013.

[3] Johannes Blömer, Richard Karp, and Emo Welzl. The rank of sparse random matrices over finite fields. *Technical report/Department of Computer Science, ETH Zurich*, 257, 1996.

[4] KA Byrd and Theresa P. Vaughan. Counting and constructing orthogonal circulants. *Journal of Combinatorial Theory, Series A*, 24(1):34–49, 1978.

[5] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[6] Danilo Gligoroski and Svein Johan Knapskog. Edon-r (256, 384, 512)–an efficient implementation of edon-r family of cryptographic hash functions. *Commentationes Mathematicae Universitatis Carolinae*, 49(2):219–240, 2008.

[7] Danilo Gligoroski, Rune Steinsmo Ødegård, Rune Erlend Jensen, Ludovic Perret, Jean-Charles Faugere, Svein Johan Knapskog, and Smile Markovski. Mqq-sig. In *International Conference on Trusted Systems*, pages 184–203. Springer, 2011.

[8] Valerii Denisovich Goppa. A new class of linear correcting codes. *Problemy Peredachi Informatsii*, 6(3):24–30, 1970.

[9] Stefan Kölbl, Martin M Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2–efficient short-input hashing for post-quantum applications. volume 2016, pages 1–29, 2017.

[10] FJ MacWilliams. Orthogonal circulant matrices over finite fields, and how to find them. *Journal of Combinatorial Theory, Series A*, 10(1):1–17, 1971.

[11] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[12] National Institute of Standards and Technology (NIST). Announcing request for nominations for public-key post-quantum cryptographic algorithms. *United States Feders Register*, 81 FR 92787, 2016.

[13] Secure Hash Standard. Fips pub 180-2. *National Institute of Standards and Technology*, 2002.

[14] D. R. Stinson. *Combinatorial Designs: Constructions and Analysis*. SpringerVerlag, 2003.

[15] Gerhard J Woeginger and Zhongliang Yu. On the equal-subset-sum problem. *Information Processing Letters*, 42(6):299–302, 1992.

[16] Zhe Zhang. *Construction of the orthogonal groups of nxn circulant matrices over finite fields*. PhD thesis, Concordia University, 1997.

A Statements

These statements “must be mailed to Dustin Moody, Information Technology Laboratory, Attention: Post-Quantum Cryptographic Algorithm Submissions, 100 Bureau Drive – Stop 8930, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930, or can be given to NIST at the first PQC Standardization Conference (see Section 5.C).”

First blank in submitter statement: full name.

Second blank: full postal address.

Third, fourth, and fifth blanks: name of cryptosystem.

Sixth and seventh blanks: describe and enumerate or state “none” if applicable.

First blank in patent statement: full name. Second blank: full postal address. Third blank: enumerate. Fourth blank: name of cryptosystem.

First blank in implementor statement: full name. Second blank: full postal address. Third blank: full name of the owner.

A.1 Statement by Each Submitter

I, ________________________, of ____, do hereby declare that the cryptosystem, reference implementation, or optimized implementations that I have submitted, known as ________________________, is my own original work, or if submitted jointly with others, is the original work of the joint submitters. I further declare that (check one):

- *I do not hold and do not intend to hold any patent or patent application with a claim which may cover the cryptosystem, reference implementation, or optimized implementations that I have submitted, known as ________________________ OR (check one or both of the following):*
  - *to the best of my knowledge, the practice of the cryptosystem, reference implementation, or optimized implementations that I have submitted, known as ________________ may be covered by the following U.S. and/or foreign patents: ____*
  - *I do hereby declare that, to the best of my knowledge, the following pending U.S. and/or foreign patent applications may cover the practice of my submitted cryptosystem, reference implementation or optimized implementations: ____*

I do hereby acknowledge and agree that my submitted cryptosystem will be provided to the public for review and will be evaluated by NIST, and that it might not be selected for standardization by NIST. I further acknowledge that I will not receive financial or other compensation from the U.S. Government for my submission. I certify that, to the best of my knowledge, I have fully disclosed all patents and patent applications which may cover my cryptosystem, reference implementation or optimized implementations. I also acknowledge and agree that the U.S. Government may, during the public review and the evaluation process, and, if my submitted cryptosystem is selected for standardization, during the lifetime of the standard, modify my submitted cryptosystem's specifications (e.g., to protect against a newly discovered vulnerability).

I acknowledge that NIST will announce any selected cryptosystem(s) and proceed to publish the draft standards for public comment.

I do hereby agree to provide the statements required by Sections 2.D.2 and 2.D.3, below, for any patent or patent application identified to cover the practice of my cryptosystem, reference implementation or optimized implementations and the right to use such implementations for the purposes of the public review and evaluation process.

I acknowledge that, during the post-quantum algorithm evaluation process, NIST may remove my cryptosystem from consideration for standardization. If my cryptosystem (or the derived cryptosystem) is removed from consideration for standardization or withdrawn from consideration by all submitter(s) and owner(s), I understand that rights granted and assurances made under Sections 2.D.1, 2.D.2 and 2.D.3, including use rights of the reference and optimized implementations, may be withdrawn by the submitter(s) and owner(s), as appropriate.

Signed:

Title:

Date:

Place:

A.2 Statement by Patent (and Patent Application) Owner(s)

If there are any patents (or patent applications) identified by the submitter, including those held by the submitter, the following statement must be signed by each and every owner, or each owner's authorized representative, of each patent and patent application identified.

I, ____________________________, of ____, am the owner or authorized representative of the owner (print full name, if different than the signer) of the following patent(s) and/or patent application(s): ________________________________

____

and do hereby commit and agree to grant to any interested party on a worldwide basis, if the cryptosystem known as __________________ is selected for standardization, in consideration of its evaluation and selection by NIST, a non-exclusive license for the purpose of implementing the standard (check one):

- *without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination, OR*
- *under reasonable terms and conditions that are demonstrably free of any unfair discrimination.*

I further do hereby commit and agree to license such party on the same basis with respect to any other patent application or patent hereafter granted to me, or owned or controlled by me, that is or may be necessary for the purpose of implementing the standard.

I further do hereby commit and agree that I will include, in any documents transferring ownership of each patent and patent application, provisions to ensure that the commitments and assurances made by me are binding on the transferee and any future transferee.

I further do hereby commit and agree that these commitments and assurances are intended by me to be binding on successors-in-interest of each patent and patent application, regardless of whether such provisions are included in the relevant transfer documents.

I further do hereby grant to the U.S. Government, during the public review and the evaluation process, and during the lifetime of the standard, a nonexclusive, nontransferrable, irrevocable, paid-up worldwide license solely for the purpose of modifying my submitted cryptosystem's specifications (e.g., to protect against a newly discovered vulnerability) for incorporation into the standard.

Signed:

Title:

Date:

Place:

A.3 Statement by Reference/Optimized Implementations' Owner(s)

The following must also be included:

I, __________________________, ____, am the owner or authorized representative of the owner __________________________ of the submitted reference implementation and optimized implementations and hereby grant the U.S. Government and any interested party the right to reproduce, prepare derivative works based upon, distribute copies of, and display such implementations for the purposes of the post-quantum algorithm public review and evaluation process, and implementation if the corresponding cryptosystem is selected for standardization and as a standard, notwithstanding that the implementations may be copyrighted or copyrightable.

Signed:

Title:

Date:

Place: