# An approach for designing fast public key encryption systems using white-box cryptography techniques

Schelkunov D., Ph.D.

ReCrypt LLC
d.schelkunov@gmail.com, schelkunov@re-crypt.com

**Abstract**

In this paper we present an approach for designing fast public key encryption cryptosystems using random primitives and error permutation. An encryption speed of such systems allows to use them for "on-the-fly" public key encryption and makes them useful for real-time communications. A small error size allows to use this approach for designing digital signature schemes

**Keywords:** public-key cryptography, white-box cryptography, digital signature, obfuscation.

## 1 Introduction

There are lot of approaches for designing asymmetric encryption schemes. Any of them is based on some NP-hard problem. The most popular and well-studied NP-hard problems are: discrete logarithm problems [1], hardness of decoding a general linear code [2], [3], lattice problems [4], [5]. The current standards of asymmetric cryptography are based on discrete logarithm problems. Unfortunately, these standards are vulnerable against Shor's algorithm [6] and a cryptographic community works on post-quantum cryptography. Promising post-quantum cryptographic schemes are lattice-based, isogeny-based or code-based.

We focus on a code-based approach and on the underlying problem of decoding a general linear code. The most known algorithm which is based on this problem is McElice cryptosystem [2]. It was the first such scheme to use randomization in the encryption process. McElice cryptosystem is a candidate for post-quantum cryptography, as it is immune to attacks using Shor's algorithm. This cryptosystem has an extremely high encryption speed and a large public key size. Unfortunately, it is not well intended for designing digital signature schemes that is the major disadvantage of such a cryptosystem.

Most of other code-based schemes, like Niederreiter one, are appeared to be vulnerable to various algebraic attacks and structural decoding [7].

In this paper we present an approach for designing fast public key encryption systems which can be used both for fast encryption and digital signature check. The approach is based on the complicity of computing decryption matrices from the obfuscated using white-box cryptography techniques [8]-[10] T-boxes. The obfuscation of a T-box consists of two secret transformations (which are the part of a secret key): concatenation with a random error vector and multiplication with a random nonsingular binary matrix. Additionally, as we can see later, source T-boxes (before obfuscation transformations) are created using random S-boxes and other random nonsingular binary matrix. These random S-boxes and binary matrix are another part of a secret key. To decrypt an encrypted message an adversary must restore binary matrices (actually, their equivalents up to linear transformations). It is equal to extracting error vectors from the T-boxes. In other words, an adversary must decode an unknown linear code.

## 2 Terminology and Notation

Let $GF(2)$ be a Galois Field of order 2, $a \cdot b$ be a product of two elements over $GF(2)$ , $a+b$ be a sum of two elements over $GF(2)$. Let $4|n$. We denote an $n$-bit vector as $\alpha^{(n)}$ and a square $n \times n$ matrix as $M^{n \times n}$. We also denote as $a^{(n)} + b^{(n)}$ a bitwise modulo-2 addition of two $n$-bit vectors $a^{(n)}$ and $b^{(n)}$.

Let $M \times \alpha$ be a product of square binary matrix $M^{n \times n}$ and $n$-bit vector $\alpha^{(n)}$ over $GF(2)$:

$$\begin{aligned} M \times \alpha &= \begin{pmatrix} m_0^0 & m_0^1 & \cdots & m_0^{n-1} \\ m_1^0 & m_1^1 & \cdots & m_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n-1}^0 & m_{n-1}^1 & \cdots & m_{n-1}^{n-1} \end{pmatrix} \times \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} = \\ &= \begin{pmatrix} m_0^0 \cdot \alpha_0 + \cdots + m_0^{n-1} \cdot \alpha_{n-1} \\ m_1^0 \cdot \alpha_0 + \cdots + m_1^{n-1} \cdot \alpha_{n-1} \\ \vdots \\ m_{n-1}^0 \cdot \alpha_0 + \cdots + m_{n-1}^{n-1} \cdot \alpha_{n-1} \end{pmatrix} \end{aligned} \tag{1}$$

In (1) $m_i^j$ - element of binary matrix $M^{n \times n}$ at the row $i$ and column $j$, $\alpha_i$ - $i$-th element (bit) of vector $\alpha^{(n)}$.

Let $t|n$ and $\frac{n}{t} = u$. Then we can split a matrix $M^{n \times n}$ to the $u^2$ square submatrices $W^{t \times t}$, a vector $\alpha^{(n)}$ to the $u$ $t$-bit subvectors $\beta_0^{(t)}, \beta_1^{(t)}, ..., \beta_{u-1}^{(t)}$

and write (1) as follows:

$$M \times \alpha = \begin{pmatrix} W_0^0 & W_0^1 & \cdots & W_0^{u-1} \\ W_1^0 & W_1^1 & \cdots & W_1^{u-1} \\ \vdots & \vdots & \ddots & \vdots \\ W_{u-1}^0 & W_{u-1}^1 & \cdots & W_{u-1}^{u-1} \end{pmatrix} \times \begin{pmatrix} \beta_0^{(t)} \\ \beta_1^{(t)} \\ \vdots \\ \beta_{u-1}^{(t)} \end{pmatrix} = \\ = \begin{pmatrix} W_0^0 \times \beta_0^{(t)} + \cdots + W_0^{u-1} \times \beta_{u-1}^{(t)} \\ W_1^0 \times \beta_0^{(t)} + \cdots + W_1^{n-1} \times \beta_{u-1}^{(t)} \\ \vdots \\ W_{u-1}^0 \times \beta_0^{(t)} + \cdots + W_{u-1}^{u-1} \times \beta_{u-1}^{(t)} \end{pmatrix} \tag{2}$$

Let $s(x) : x^{(t)} \to z^{(t)}$ be a bijective nonlinear transformation (S-box) where $t$ is a bit size of the vectors $x$ and $z$. By replacing $\beta_0^{(t)}, \beta_1^{(t)}, ..., \beta_{u-1}^{(t)}$ with $s_0(x_0), s_1(x_1), ..., s_{u-1}(x_{u-1})$ in (2) we get the following:

$$F(x_0, x_1, ..., x_{u-1}) = \begin{pmatrix} W_0^0 & W_0^1 & \cdots & W_0^{u-1} \\ W_1^0 & W_1^1 & \cdots & W_1^{u-1} \\ \vdots & \vdots & \ddots & \vdots \\ W_{u-1}^0 & W_{u-1}^1 & \cdots & W_{u-1}^{u-1} \end{pmatrix} \times \begin{pmatrix} s_0(x_0) \\ s_1(x_1) \\ \vdots \\ s_{u-1}(x_{u-1}) \end{pmatrix} = \\ = \begin{pmatrix} W_0^0 \times s_0(x_0) + \cdots + W_0^{u-1} \times s_{u-1}(x_{u-1}) \\ W_1^0 \times s_0(x_0) + \cdots + W_1^{n-1} \times s_{u-1}(x_{u-1}) \\ \vdots \\ W_{u-1}^0 \times s_0(x_0) + \cdots + W_{u-1}^{u-1} \times s_{u-1}(x_{u-1}) \end{pmatrix} \tag{3}$$

From the right side of (3) follows:

$$F(x_0, x_1, ..., x_{u-1}) = \begin{pmatrix} W_0^0 \times s_0(x_0) \\ W_1^0 \times s_0(x_0) \\ \vdots \\ W_{u-1}^0 \times s_0(x_0)) \end{pmatrix} + \cdots \\ \cdots + \begin{pmatrix} W_0^{u-1} \times s_{u-1}(x_{u-1}) \\ W_1^{u-1} \times s_{u-1}(x_{u-1}) \\ \vdots \\ W_{u-1}^{u-1} \times s_{u-1}(x_{u-1})) \end{pmatrix} = \\ = T_0(x_0) + \cdots + T_{u-1}(x_{u-1}) \tag{4}$$

The functions $T_i(x_i) : x_i^{(t)} \to \lambda_i^{(n)}$ in (4) are called T-boxes. Every T-box is a lookup table function. We can combine the T-boxes as follows:

$$F(x_0, x_1, ..., x_{u-1}) = T_0^c(x_0, x_1) + \cdots + T_{\frac{u}{2}-1}^c(x_{u-1}, x_{u-1}) \tag{5}$$

In (5)

$$T_i^c(x_k, x_l) = \begin{pmatrix} W_0^k \times s_k(x_k) + \cdots + W_0^l \times s_l(x_l) \\ W_1^k \times s_k(x_k) + \cdots + W_1^l \times s_l(x_l) \\ \vdots \\ W_{u-1}^k \times s_k(x_k)) + \cdots + W_{u-1}^l \times s_l(x_l) \end{pmatrix} \tag{6}$$

## 3 Private and public keys

At the first step we generate a set $S = \{s_0, s_1, \cdots, s_{u-1}\}$, $s_i(x) : x^{(t)} \rightarrow z^{(t)}$ of $u$ $t$-bit s-boxes in the random way using, for example, Chaos theory [11] - [13]. After that we (randomly) generate a nonsingular binary matrix $M^{n \times n}, n = u \cdot t$. Then we select error size $es$ and randomly generate a nonsingular binary matrix $H^{h \times h}$ , where $h = n + es$. A tuple $\{S, M, H\}$ is a private key.

Having a private key we generate a set of combined T-boxes (5). After that we construct a lookup function $T_i^{ex}(\alpha^{(t)}, \beta^{(t)}) : \{\alpha^{(t)}, \beta^{(t)}\} \rightarrow z^{(h=n+es)}$ from $T_i^c(\alpha^{(t)}, \beta^{(t)})$ by expanding the result of every $T_i^c(\alpha^{(t)}, \beta^{(t)})$ by $es$ bits. Then we fill $es$ high bits of the result of every $T_i^{ex}(\alpha^{(t)}, \beta^{(t)})$ with generated using PRNG $es$-bit values $err_{i,\alpha,\beta} = err_i(\alpha^{(t)}, \beta^{(t)})$ (Figure 1). These values must satisfy the following conditions:

$$\forall i \sum_{\alpha^{(t)}, \beta^{(t)}} err_{i,\alpha,\beta} = 0 \tag{7}$$

$$err_{i_1,\alpha_1,\beta_1} = err_{i_2,\alpha_2,\beta_2} \implies i_1 = i_2, \alpha_1 = \alpha_2, \beta_1 = \beta_2 \tag{8}$$

After that mix the bits of the result of every $T_i^{ex}(\alpha^{(t)}, \beta^{(t)})$ in the following way (Figure 2):

$$T_i^{mix}(\alpha^{(t)}, \beta^{(t)}) = H^{h \times h} \times T_i^{ex}(\alpha^{(t)}, \beta^{(t)}) \tag{9}$$

The set of mixed T-box-es $\{T_i^{mix}, i \in [0, \frac{u}{2} - 1]\}$ (which are determined as lookup tables) is a public key.

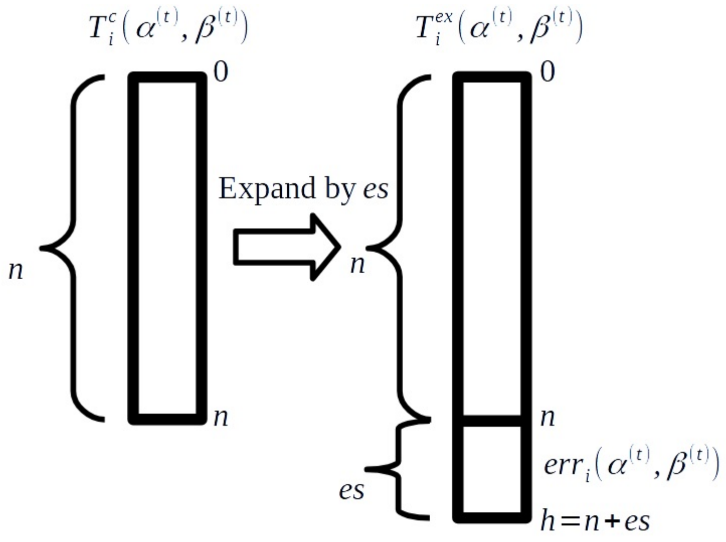


Figure 1. Expanding of the result of a T-box by error vector

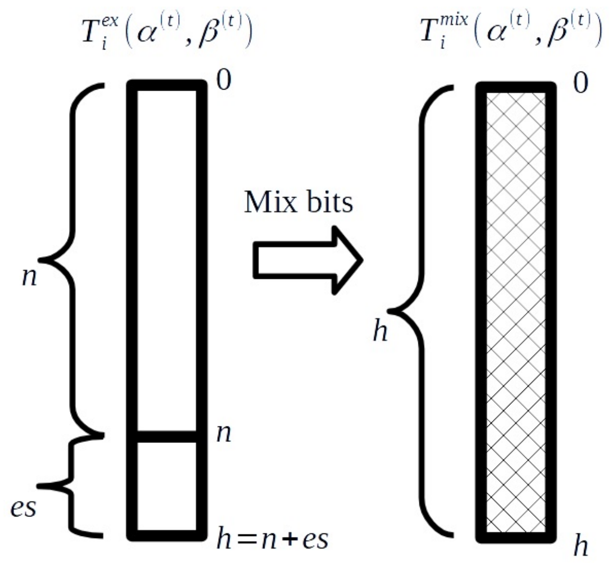


Figure 2. Mixing bits of the result of a T-box

## 4 Encryption with a public key

Let $x^{(n)} = \{x_0^{(t)}, x_1^{(t)}, \cdots, x_{u-1}^{(t)}\}$ be an $n$-bit source message. We encrypt it in the following way:

$$c = Encr(x) = T_0^{mix}(x_0^{(t)}, x_1^{(t)}) + \cdots + T_{\frac{u}{2}-1}^{mix}(x_{u-2}^{(t)}, x_{u-1}^{(t)}), \tag{10}$$

where $c$ is a $h$-bit encrypted message.

# 5 Decryption with a private key

As we mentioned above, a tuple $\{S, M, H\}$ is a private key. At the first step we calculate inverse matrices $H'^{h\times h}, M'^{n\times n} : H'^{h\times h} \times H^{h\times h} = I^{h\times h}, M'^{n\times n} \times M^{n\times n} = I^{n\times n}$ ($I^{h\times h}, I^{n\times n}$ are identity matrices) and inverse S-box-es $S' = \{s'_0, s'_1, \cdots, s'_{u-1}\} : s'_i(s_i(x)) = x = s_i(s'_i(x))$. After that we multiply an input $h$-bit ciphertext $c$ with $H'^{h\times h}$:

$$dmx^{(h)} = Demix(c^{(h)}) = H'^{h\times h} \times c^{(h)} \tag{11}$$

High $es$ bits of $dmx$ (Figure 3) contain a summary error:

$$err = err_0(x_0^{(t)}, x_1^{(t)}) + \cdots + T_{\frac{u}{2}-1}(x_{u-2}^{(t)}, x_{u-1}^{(t)}) \tag{12}$$

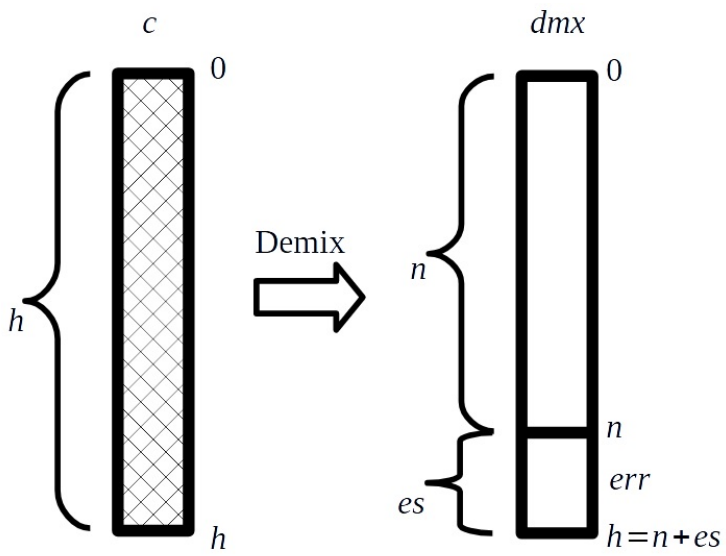


Figure 3. "Demix" function

So, we can reduce a size of $dmx$ from $h$ to $n$ by cutting the high $es$ bits. Now we have an error-free $n$-bit vector $ef^{(n)}$ (Figure 4):

$$ef^{(n)} = Reduce(dmx^{(h)}) \tag{13}$$

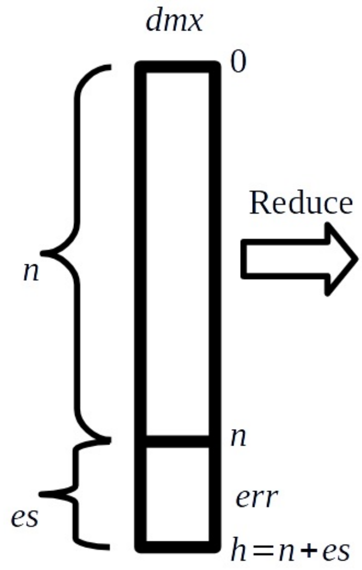


Figure 4. "Reduce" function

After that we multiply $M'^{n\times n}$ with $ef^{(n)}$ and get a $n$-bit vector $z^{(n)}$ :

$$z^{(n)} = M'^{n\times n} \times ef^{(n)} \tag{14}$$

We can represent a $n$-bit vector $z^{(n)}$ as a vector with $u$ $t$-bit coordinates $z^{(n)} = \{z_0^{(t)}, z_1^{(t)}, \cdots, z_{u-1}^{(t)}\}$ . So, to get a source message $z^{(n)}$ we apply inverse S-box-es in the following way:

$$x^{(n)} = \{x_0^{(t)}, \cdots, x_{u-1}^{(t)}\} = \{s'_0(z_0^{(t)}), \cdots, s'_{u-1}(z_{u-1}^{(t)})\} \tag{15}$$

# 6 Digital signature scheme

Let us briefly remind ourselves a typical digital signature algorithm. Let $Hash(m) : m \to y^{(h)}$ be a hash function, $Encr(y, private\_key) : y \to x^{(n)}$ be a function that encrypts some input vector with private key, $Decr(x, public\_key) : x \to y^{(h)}$ be a function that decrypts some input vector with public key. To sign a message $m$ Alice calculates its hash function and then encrypts the result with her private key. When sending a message to Bob she attaches to it an encrypted with her private key hash: $m\|sgn, sgn = Encr(Hash(m), private\_key)$. After receiving a signed message $m\|sgn$ from Alice Bob calculates a hash of $m$ and compares the result with $Decr(sign, public\_key)$, where $public\_key$ is a public key of Alice. The signature is valid if $Hash(m) = Decr(sign, public\_key)$.

In our approach a size of an error vector is $es$ bits and $es = h - n$. As we can see, there are $2^n$ solutions of (10) in the space of $h$-bit vectors. In other words, every $h$-bit vector is a solution of (10) with a probability of $\frac{1}{2^{es}}$.

So, we can use the following digital signature algorithm:

1. Create a $es$-bit vector $cnt$ and initialize it with 0.
2. Concatenate a source message $src$ with a counter $cnt$: $m = src \| cnt$.
3. Calculate a $h$-bit hash of m: $hsh = Hash(m) : m \to hsh^{(h)}$.
4. Decrypt $hsh$ with a private key: $sgn^{(n)} = Decr(hsh, private\_key) : hsh^{(h)} \to sgn^{(n)}$.
5. Encrypt calculated at the previous step $sgn$ with a public key: $dh^{(n)} = Encr(sgn, public\_key) : sgn^{(n)} \to dh^{(h)}$.
6. Compare $dh$ and $hsh$. If they are not equal, increment $cnt$ and repeat the steps from 2 to 6.
7. Concatenate $m$ with $sgn$: $ms = m \| sgn$.

So, $n$-bit vector $sgn$ is a signature of a source message $src$.

# 7 Parameters

To get a private key from a public one an adversary must firstly eliminate errors from the results of T-boxes $T_i^{mix}$. Every this result is obfuscated by the matrix H (which is a part of a private key). We can write (9) as follows:

$$T_i^{mix}(\alpha^{(t)}, \beta^{(t)}) = H^{h \times h} \times T_i^{ex0}(\alpha^{(t)}, \beta^{(t)}) + H^{h \times h} \times exterr_i(\alpha^{(t)}, \beta^{(t)}), \quad (16)$$

where $T_i^{ex0}(\alpha^{(t)}, \beta^{(t)})$ is the same as $T_i^{ex}(\alpha^{(t)}, \beta^{(t)})$ , but high $es$ bits of the result are zero, $exterr_i(\alpha^{(t)}, \beta^{(t)})$ returns a $h$-bit vector, where low $n$ bits are zero and high $h - n$ bits are equal to the result of $err_i(\alpha^{(t)}, \beta^{(t)})$. A space of $h$-bit vectors $rev_{i,\alpha,\beta} = H^{h \times h} \times exterr_i(\alpha^{(t)}, \beta^{(t)})$ makes it hard to restore linear relationship between sub-vectors of the results of T-box-es. In other words, having a set of all of the results of $T_i^{mix}(\alpha^{(t)}, \beta^{(t)})$ , it is hard to build an inverse binary $h \times h$ matrix which is necessary to decrypt encrypted messages and to restore a private key from a public one.

For practical implementation we recommend the following parameters: $n$=256 bits, $h$=272 bits, $es$=16 bits, $t = 4$ bits.

# 8 An underlying hard problem

Firstly an adversary can brutforce $x^{(n)}$ to get the appropriate ciphertext $c$ with a complexity about $O(2^n)$. For the recommended $n$ this complexity is $O(2^{256})$.

Let the result of the every of $T_i^{mix}(\alpha^{(t)}, \beta^{(t)})$ be a $h$-bit binary vector $\zeta_j^{(h)}, j \in [0, \frac{2^{2\cdot t}\cdot u}{2} - 1]$. An attack to the our approach could be the same as one to the generic rucksack cryptosystem. Let we have two set of integers $I \subset \{i : 0 \leq i \leq \frac{2^{2\cdot t}\cdot u}{4} - 1\}$ and $J \subset \{j : \frac{2^{2\cdot t}\cdot u}{4} \leq i \leq \frac{2^{2\cdot t}\cdot u}{2} - 1\}$. Then we can compute and make a list of the values $A_I = \sum\limits_{i \in I} \zeta_i$ and $B_J = c - \sum\limits_{j \in J} \zeta_j$. These lists include a pair of sets $I_0$ and $J_0$ satisfying $A_{I_0} = B_{J_0}$, and the sets $I_0$ and $J_0$ give a solution to the problem:

$$c = \sum_{i \in I_0} \zeta_i + \sum_{j \in J_0} \zeta_j \tag{17}$$

The complexity of this algorithm is about $O(2^{\frac{2^{2\cdot t}\cdot u}{4}})$ which is more than $O(2^n)$.

From (10) we can construct the following binary matrix:

$$L^{(\frac{2^{2\cdot t}\cdot u}{2}+h)\times(\frac{2^{2\cdot t}\cdot u}{2}+1)} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ \zeta_0 & \zeta_1 & \zeta_2 & \cdots & \zeta_{\frac{2^{2\cdot t}\cdot u}{2}-1} & c \end{pmatrix} \tag{18}$$

The submatrix $E^{\frac{2^{2\cdot t}\cdot u}{2}\times\frac{2^{2\cdot t}\cdot u}{2}}$ of (18) (first $\frac{2^{2\cdot t}\cdot u}{2}$ rows $\frac{2^{2\cdot t}\cdot u}{2}$ columns) is an identity one. So, the columns of the binary matrix (18) form a basis of the lattice of binary vectors or a basis of the linear code over $GF(2)$. From (10) it follows that some linear combination over $GF(2)$ of binary vectors $\zeta_j^{(h)}$ gives the binary vector $c$:

$$\sum_{j=0}^{\frac{2^{2\cdot t}\cdot u}{2}-1} \mu_j \cdot \zeta_j^{(h)} + c = 0, \mu_j \in GF(2), \tag{19}$$

where $\mu_j$ is an element of a binary vector $\mu^{(\frac{2^{2\cdot t}\cdot u}{2})}$ on the position j. From (18) and (19) we get:

$$\sum_{j=0}^{\frac{2^{2\cdot t}\cdot u}{2}-1} \mu_j \cdot L^j + L^{\frac{2^{2\cdot t}\cdot u}{2}} = \psi^{(\frac{2^{2\cdot t}\cdot u}{2}+h)}, \mu_j \in GF(2), \tag{20}$$

where $L^j$ is a $j$-th column of the binary matrix $L$, $\psi^{(\frac{2^{2\cdot t}\cdot u}{2}+h)}$ is a binary vector (codeword). As we can see, the coordinates of nonzero bits of $\psi$ are equal to the coordinates $j$ of nonzero elements $\mu_j$ of $\mu$ and vice versa. If we know a binary vector $\psi$ we can easy decrypt an encrypted message $c$ by matching its coordinates with appropriate T-box-es. Note that $\psi$ is a low weight vector (codeword) with a Hamming weight $wt(\psi) = wt(\mu) = \frac{u}{2}$. So, the problem of finding the binary vector $\psi$ from the code (18) is the problem of finding low weight codewords which is known to be NP-hard [14][15].

The practical experiments show that the reduction techniques for binary codes including LLL [15] are not effective in finding codeword $\psi$ from (20).

# 9 References


[1] Blake, Ian F.; Garefalakis Theo, "On the complexity of the discrete logarithm and Diffie–Hellman problems", *Journal of Complexity. Festschrift for Harald Niederreiter, Special Issue on Coding and Cryptography*, **20 (2)**, 2004, 148–170.

[2] McEliece, Robert J., "A Public-Key Cryptosystem Based on Algebraic Coding Theory", *DSN Progress Report*, **44**, 1978, 114–116.

[3] Dinh H.; Moore C.; Russell A., "McEliece and Niederreiter cryptosystems that resist quantum Fourier sampling attacks", *Advances in cryptology—CRYPTO 2011. Lecture Notes in Computer Science*, **6841**, ed. Rogaway P., Springer, Berlin, Heidelberg, 2011, 761–779.

[4] Ajtai M., "Generating hard instances of lattice problems", *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, Philadelphia, Pennsylvania, United States, 1996, 99–108.

[5] Ajtai M., "The shortest vector problem in L2 is NP-hard for randomized reductions", *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, ACM, Dallas, Texas, United States, 1998, 10–19.

[6] Shor P., "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Journal on Computing*, **26 (1997)**, 1997, 1484–1509.

[7] V. M. Sidelnikov, S. O. Shestakov, "On the insecurity of cryptosystems based on generalized Reed-Solomon codes", *Discrete Mathematics and Applications*, **2 (4)**, VSP, 1992, 439–444.

[8] S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, "White-Box Cryptography and an AES Implementation", In 9th Annual Workshop on Selected Areas in Cryptography (SAC 2002), 2002.

[9] B. Wyseur, "White-Box Cryptography, PhD thesis", 2009.

[10] D. Schelkunov, "White-Box Cryptography and SPN ciphers. LRC method", *Cryptology ePrint Archive: Report 2010/419*, 2010.

[11] Goce Jakimoski and Ljupˇco Kocarev, "Chaos and Cryptography: Block Encryption Ciphers Based on Chaotic Maps", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: FUNDAMENTAL THEORY AND APPLICATIONS*, **48 (2)**, 2001.

[12] M. Asim, V. Jeoti, "Efficient and simple method for designing chaotic s-boxes", *ETRI Journal*, **30 (1)**, 2008, 170 - 172.

[13] G. Jakimoski, L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **48 (2)**, 2001.

[14] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg, "On the Inherent Intractability of Certain Coding Problems", *IEEE Transactions on Information Theory 24 (1978)*, 1978, 384–386.

[15] Debris-Alazard, Thomas; Ducas, Leo; Woerden, Wessel, "An Algorithmic Reduction Theory for Binary Codes: LLL and more", *IEEE Transactions on Information Theory*, **PP(99)**, 2022.