Building the dynamic diffusion layer for SPN block ciphers based on direct exponent and scalar multiplication

Tran Thi Luong

***Abstract*—Maximum Distance Separable (MDS) matrices have been applied not only in coding theory but also in the design of block ciphers and hash functions. In this paper, we propose algorithms for building a dynamic diffusion layer for SPN block ciphers based on the direct exponent and scalar multiplication. The proposed dynamic algorithms contribute to improving the security of SPN block ciphers against strong attacks on block ciphers such as linear attacks, differential attacks.**

***Tóm tắt*— Ma trận phân tách cực đại (MDS) không chỉ được áp dụng trong lý thuyết mã hóa mà còn trong việc thiết kế mật mã khối và các hàm băm. Trong bài báo này, tác giả đề xuất các thuật toán để xây dựng một lớp khuếch tán động cho các mật mã khối SPN dựa trên phép nhân và lũy thừa trực tiếp. Các thuật toán động được đề xuất góp phần nâng cao tính an toàn của mật mã khối SPN trước các loại tấn công mạnh vào mật mã khối như tấn công tuyến tính, tấn công vi sai.**

Keywords—MDS matrix, dynamic block cipher.

Từ khóa—Ma trận MDS, mã khối động.

I. INTRODUCTION

In cryptography, confusion and diffusion are two indispensable properties of a secure cipher. The simplest way to achieve these properties is a substitution-permutation network. This network takes as input a plaintext block and a secret key and applies many transformation "rounds" or "layers" of substitution boxes (S-boxes) and permutation boxes (P-boxes) to create cipher text block.

MDS matrices play a very important role in block cipher design, especially substitution-permutation network (SPN). Therefore, they have been used for many ciphers today. To improve the security of block ciphers, there are some methods for making dynamically these ciphers such asmake dynamically at the substitution layer, at the diffusion layer or both.

For the method making dynamically at diffusion layer, there are some works in the literature about this direction. In [3] and [8], the authors generated key-dependent MDS matrices for each round of encryption to build a key-dependent diffusion layer. In [3], permutation and scalar multiplication [1] of the rows of the matrix are used, where scalar multiplication and permutation are generated from a secret key and a random bit generator function. In [8] dynamic MDS matrices are also generated using scalar multiplication [1] for each round, where a secret key and a random bit generator are generated by the scalar multiplication. In [9], a dynamic block cipher was proposed. This cipher is made dynamically at both substitution and permutation layers by a bank of substitution boxes and key-dependent MDS matrices.

In addition, there are some other methods for a block cipher with a dynamic diffusion layer in the literature. In [19], the authors proposed a dynamic block cipher with a variable size block encryption algorithm using a dynamic-key mechanism. This cipher was designed with an unlimited size of the key, a dynamic key-dependent permutation and a changing size of the encryption block in each round. In [22], a dynamic SPN block cipher was proposed based on AES (denoted DRAES). In which, a rotation transform is made dynamically where an amount for rotating depends on the data (plaintext and ciphertext) in AddRoundKey and depends on the key in the key extension of AES. In [24], the AES block cipher was made dynamically at the diffusion layer. In particulaly, instead of using a static MixColumn transformation, the approach is to use a dynamic MixColumn transformation based on key-dependent DNA structures and processes. In [30], the authors proposed two methods of increasing the security of AES block ciphers: using the dynamic MixColumns

transformation which uses a dynamic MDS matrix by a 3D chaos mapping.

In this paper, we propose algorithms to build a dynamic diffusion layer for SPN block ciphers based on the direct exponent and scalar multiplication. These transformations are capable of preserving the good cryptographic properties of the MDS matrix when made dynamically [4, 6, 7]. In [3, 8], the authors also used these transformations but the algorithms in [3, 8] are either quite complex or do not mention the problem of preserving the good cryptographic properties of the MDS matrix when made dynamically. Therefore, our proposed dynamic algorithms will contribute to improving the security of the SPN block cipher against strong attacks on the block ciphers such as the linear attack and differential attack.

The paper is organized as follows. In Section 2, preliminaries and related works are introduced. Section 3 presents algorithms for building a dynamic diffusion layer for SPN block ciphers based on the direct exponent and scalar multiplication. Section 4 includes some security analysis for dynamic SPN block ciphers after applying the above algorithms. And conclusions of the paper are in Section 5.

II. PRELIMINARIES AND RELATED WORKS

C. MDS matrix

The MDS matrices provide the property of perfect diffusion so they have useful applications in block ciphers and hash functions. MDS matrices come from coding theory with maximal distance separable codes. In coding theory, there are the following important theorems about MDS matrices:

Theorem 1 ([10, page 33]). *If C is a $[n, k, d]$ code then $n - k \geq d - 1$.*

Codes with $n - k = d - 1$ (or $d = n - k + 1$) are called maximum distance separable code, or MDS code for short.

Theorem 2 ([10, page 321]). *A $[n, k, d]$ code C with generator matrix $G = [I|A]$, where A is a $k \times (n - k)$ matrix, is MDS if and only if every square submatrix (formed from any i rows and any i columns, for any $i = 1, 2, \dots, min\{k, n - k\}$) of A is nonsingular.*

D. Direct exponent transformation

The definition of a direct exponent of an MDS matrix was introduced by Ghulam Murtaza and Nassar Ikram [1]. The authors gave the direct exponent definition, as follows:

Definition 2 [1]. *Let F be a Galois field. Let matrix $A = [a_{i,j}]_{m\times m}$, $a_{i,j} \in F$, then $A_{d^e} = [a_{i,j}^e]_{m\times m}, (e = 1, 2, 3 \dots)$ is called direct e exponent matrix of A. And A_{d^2} is called a direct square matrix of A.*

In [4], we showed that the direct exponent transformation is capable of preserving many of the good cryptographic properties of the MDS matrix, such asMDS, Involutory, Symmetric, Recursive, the number of 1's and distinct elements in the matrix, Circulant and circulant-like.

In [5], the cycle (τ) of the direct exponent of an MDS matrix was shown.

In [6], we also showed that the direct exponent is also able to preserve the number of fixed points and coefficient of fixed points [11] of the MDS matrix.

E. Scalar multiplication

Ghulam Murtaza and Nassar Ikram [1] defined the scalar multiplication for the MDS matrix as follows:

Let $A = [A_1, \dots, A_m]^T$ be an MDS matrix and $A_i = \begin{bmatrix} a_{i,1} & \cdots & a_{i,n} \end{bmatrix}$, $a_{i,j} \in F_q$. Denote vector $E = [e_i]$, $i = 1,2, \dots, m$ where $e_i \neq 0 \in F_q$, $i = 1, 2, \dots, m$. Then the scalar multiplication of E and A generates an MDS matrix denoted by: $EA = [e_1A_1 \ \dots \ e_mA_m]^T$ where $e_iA_i = \begin{bmatrix} e_ia_{i,1} & \cdots & e_ia_{i,n} \end{bmatrix}$.

In [7], we expand the definition of scalar multiplication given by the authors in [1]. Let $A = [a_{i,j}]_{m\times m}$, $a_{i,j} \in GF(p^r)$ be an MDS matrix. Denote vectors $E = (e_1, e_2, \dots, e_m)$, $F = (f_1, f_2, \dots, f_m)$, (where $e_i, f_i \in GF^*(p^r)$). Then the scalar multiplication of E, F and A generates an MDS matrix denoted by: $(E, F)(A) = [b_{i,j}]_{m\times m}$ where $b_{i,j} = e_if_ja_{i,j}$.

We showed that the scalar multiplication is capable of preserving the involutory property of MDS matrices.

In [7], we also showed the cycle of scalar multiplication.

III. PROPOSING ALGORITHMS TO BUILD DYNAMIC DIFFUSION LAYERS FOR BLOCK CIPHER

Suppose that it is necessary to build a dynamic diffusion layer for a SPN block cipher consisting of T rounds. Assume that the encryption key is of length n bits where n must be large enough that a key brute force attack cannot perform, for example, the encryption key in AES has lengths of 128, 192, or 256 bits.

Here, two transformations are used: direct exponent (denoted by T_p) or scalar multiplication (denoted by T_M).

Dynamic MDS matrices that are dependent on a given secret key of length n ($n \geq 128$) are created. Assume that the SPN block cipher is performed over the field $GF(2^r)$. Two following dynamic algorithms for building the dynamic diffusion layer of the SPN block cipher are proposed.

A. *Dynamic algorithm 1: each secret key generates only one new MDS matrix used for every round*

Description

Algorithm idea: In this dynamic algorithm, assume that there is a given MDS matrix M and a secret key K, MDS matrix transformations are used to generate a new MDS matrix from the original matrix and depend on that secret key. This new key-dependent MDS matrix will be used in the diffusion layer for every round of the encryption process. In this algorithm, the first key bit of the key K is used to determine which matrix transformation is used (in practice, it is possible to increase the key by one bit at the beginning so as not to affect the key bits used for encryption/decryption process).

For the direct exponent T_p and the MDS matrix M, it is possible to find a cycle τ ($1 < \tau \leq r$) of direct 2^k ($0 \leq k \leq \tau - 1$) exponent of this matrix. Of course, one should choose a matrix M that has the cycle $\tau = r$ (maximum) and has good cryptographic properties (see Theorem 3).

Algorithm 1.

INPUT: A secret key K, An MDS matrix M of size m over $GF(2^r)$.

OUTPUT: An MDS matrix M_K generated from M and K. The matrix M_K will be used in the diffusion layer for every round of the encryption process.

Detailed steps are as follows:

Step 1: Take the first bit of the key K to determine which matrix transformation is used in this algorithm:

If it is 0, then choose the transformation as T_p.

If it is 1, then choose the transformation as T_M.

Step 2:

If the result of step 1 is T_p then execute:

+ Step 2.1. Get the next n_r bits of K (and b is the corresponding number), where n_r is the bit length in the binary representation of r. Take $l = b \bmod r$.

+ Step 2.2. Find the direct 2^l exponent matrix of the original M matrix, obtain $M_K = M_{d^{2^l}}$.

If the result of step 1 is T_M then execute:

+ Step 2.3. Take consecutively non-zero bit segments (each has r bits) of the key K (i.e., if there is a segment where its r bits are all zero, right rotate by one bit until you get the segment where its r bits are not concurrently zero). Take m such segments. Convert those r-bit segments to a non-zero element $e_i \in GF(2^r)$, $0 \leq i \leq m - 1$.

+ Step 2.4. Find the elements $f_i = e_i^{-1} \in GF(2^r)$, $(0 \leq i \leq m - 1)$ (make $e_i f_i = 1$ satisfy the condition of the Theorem 5 to

preserve the involutory property of M if M is an involutory MDS matrix).

+ Step 2.5. For $0 \le i \le m-1$, multiply all the elements of row i of M by the element e_i, and multiply all the elements of column i of M by the element f_i over $GF(2^r)$. Then, we get a new MDS matrix, denoted by M_K.

The result of the algorithm: is the matrix M_K, this matrix will be used for the diffusion layer in every round of the encryption process. This matrix M_K preserves some good properties of the original matrix M (depending on the transformation that is used).

Encryption and decryption process

Both parties participating in communication will keep publicly and secretly as follows:

Public: The encryption/decryption algorithm, the original MDS matrix M (of size m), and the dynamic algorithm 1 (Algorithm 1).

Secret: The secret key K.

Before implementing the encryption/decryption for any message, both parties must perform Algorithm 1 with the secret key K and the original MDS matrix to create the dynamic MDS matrix M_K. In addition, they need to calculate the inverse matrix of M_K to serve the decryption process (unless M is involutory).

After that, the two participants could start the communication. They will encrypt messages using the SPN algorithm with the matrix M_K used in the diffusion layer for every round of the encryption process. The decryption process is done with the inverse matrix of M_K.

Later when the parties want to change the secret key with another key, they must perform Algorithm 1 again to create a new MDS matrix depending on the new key.

B. Dynamic algorithm 2: each secret key generates two new MDS matrices used alternating in rounds

Description

Algorithm idea: In this dynamic algorithm, assume that there are two given MDS matrices A, B and a secret key K, MDS matrix transformations are used to generate two new MDS matrices from the original matrix and depend on that secret key. MDS matrix transformations can be T_p or T_M. Two new MDS matrices dependent on the secret key will be used alternating in the diffusion layer for every round in the encryption process. In this algorithm, the first two key bits of the key K are used to determine which matrix transformation is used (in practice, it is possible to increase the key by two bits at the beginning so as not to affect the key bits used for encryption/decryption process).

Algorithm 2.

INPUT: A secret key K, two MDS matrices A, B of size m over $GF(2^r)$.

OUTPUT: Two MDS matrices A_K, B_K generated from A, B and K. The two matrices A_K, B_K will be used alternating in the diffusion layer for every round in the encryption process.

Detailed steps are as follows:

Step 1: Take the first two bits of the key K to determine which matrix transformation is used in this algorithm:

If they are 10, then choose the transformation as T_p.

If they are 01, then choose the transformation as T_M.

If they are 00 or 11, then choose both T_p and T_M.

Step 2:

If the result of step 1 is T_p then execute:

+ Step 2.1. Get the next n_r bits of K (and b is the corresponding number), where n_r is the bit length in the binary representation of r. Take $l_1 = b \bmod r$. Get n_r last bits of K (and b' is the corresponding number) and determine $l_2 = b' \bmod r$.

+ Step 2.2. Find the direct 2^{l_1} exponent matrix of the original A matrix, obtain $A_K =$

$A_{d^{2^{l_1}}}$, and the direct 2^{l_2} exponent matrix of the original B matrix, obtain $B_K = A_{d^{2^{l_2}}}$.

If the result of step 1 is T_M then execute:

+ Step 2.3. Take the next r bits non-concurrently equal to 0 of K and switch the r bits to element $e_1 \in GF(2^r)$. So $e_1 \neq 0 \in GF(2^r)$. If the above r bits are all equal to 0, then right rotate by 1 bit of K until obtain r bits for $e_1 \neq 0$. Take the last r bits of K and switch this r bits to element $e_2 \in GF(2^r)$. If $e_2 = 0 \in GF(2^r)$, then left rotate by 1 bit until obtain element $e_2 \neq 0 \in GF(2^r)$.

+ Step 2.4. Find the elements $f_1 = e_1^{-1} \in GF(2^r)$ and $f_2 = e_2^{-1} \in GF(2^r)$ (make $e_1 f_1 = 1$ and $e_2 f_2 = 1$, satisfy the condition of the Theorem 5 to preserve the involutory property of A, B if A, B are involutory MDS matrices).

+ Step 2.5. Take the next n_m bits at the beginning of K (after step 2.3), where n_m is the bit length in the binary representation of m.

Then convert this string to an integer, denoted by b, taking $i = b \bmod m$, so $0 \leq i \leq m-1$. Get the next n_m bits (by the right-to-left direction) at the end of the K bit sequence (after step 2.3). Then convert this string to an integer, denoted by b', taking $j = b' \bmod m$, thus $0 \leq j \leq m-1$. We specify that the rows and columns of the matrix A, B will be numbered from 0 to $m-1$.

+ Step 2.6. Multiplying all elements of row i of A by element e_1, and all elements of column i of A by element f_1 over field $GF(2^r)$, we obtain a new matrix, denoted by A_K. Next, multiplying all row j elements of B by element e_2, and all elements of column j of B by element f_2 over field $GF(2^r)$, we obtain a new matrix, denoted by B_K.

If the result of step 1 is both T_p and T_M then (apply T_p to A and T_M to B):

+ Step 2.7. Take the next n_r bits of K (and b is the corresponding number), where n_r is the bit length in the binary representation of r. Take $l = b \bmod r$. Take the last r bits of K and convert this string of r bits to the element $e \in GF(2^r)$. If $e = 0 \in GF(2^r)$ then left rotate 1 bit until obtain element $e \neq 0 \in GF(2^r)$.

+ Step 2.8. Find the direct 2^l exponent matrix of the original matrix A, yielding $A_K = A_{d^{2^l}}$. Find the element $f = e^{-1} \in GF(2^r)$.

+ Step 2.9. Take the next n_m bits at the beginning of K (after step 2.7), where n_m is the bit length in the binary representation of m. Then convert this string to an integer, denoted by b, taking $i = b \bmod m$, so $0 \leq i \leq m-1$.

+ Step 2.10. Multiplying all elements of row i of B by element e, and all elements of column i of B by element f over field $GF(2^r)$, we obtain a new matrix, denoted by B_K.

The results of the algorithm: are two matrices A_K, B_K. The two matrices will be used alternating in the diffusion layer for every round in the encryption process. For example, matrix A_K is used in odd rounds such as1, 3, 5, ...; matrix B_K will be used in even rounds like 2, 4, 6, …

Note. In Algorithm 2, if we want the T_M transformation to have the largest cycle as possible (≤ 255) by the key K, then in step 2.7, we need to traverse e by the secret key K. For each e, element $f = e^{-1} \in GF(2^r)$ will be calculated, as well $ord(e), ord(f)$. Then, in order to make the cycle of T_M as large as possible by the key K, we will choose e by the key K such that $ord(e)$ or $ord(f)$ is the largest among the elements e found from K. In fact, in step 2.10, we use two vectors: E is a vector with only the i^{th} component equal to e, the rest is all equal to 1; F is a vector with only the i^{th} component equal to f, the rest is all equal to 1.

Encryption and decryption process

Both parties participating in communication will keep publicly and secretly as follows:

Public: The encryption/decryption algorithm, the two original MDS matrices A, B (of size m), and the dynamic algorithm 2 (Algorithm 2).

Secret: The secret key K.

Before implementing the encryption/decryption for any message, both parties must perform Algorithm 2 with the secret key K and the original MDS matrices to create two dynamic MDS matrices A_K, B_K. In addition, they need to calculate two inverse matrices of A_K anh B_K to serve the decryption process.

After that, the two participants could start the communication. They will encrypt messages using the SPN algorithm with the two matrices A_K and B_K used alternating in the diffusion layer for every round in the encryption process. The decryption process is done with the inverse matrices of A_K and B_K.

Later when the parties want to change the secret key with another key, they must perform Algorithm 2 again to create a new MDS matrix depending on the new key.

IV. SECURITY ANALYSIS OF DYNAMIC ALGORITHMS

We analyze the ability of SPN block ciphers after using the above dynamic algorithms against strong attacks on block ciphers such as differential and linear attacks.

Linear attack [12, 14] is a known-plaintext attack, that requires a large number of plaintext and ciphertext pairs corresponding to a key unknown to find. Differential attack [12, 13] is a chosen-ciphertext attack based on collecting ciphertexts created from given plaintexts. Both of these attacks are based on knowing each element in the structure of the block cipher. When we use matrix transformations to make the diffusion layer dynamic, the cryptanalysts will lose information about the diffusion layer of the block cipher, and the complexity of cryptanalysis is increasing. Thus, the linear and differential attack will be much more difficult to do because of dynamic MDS matrices.

More specifically, for the proposed algorithms, although the original MDS matrices are public, the problem is that cryptanalysts don't know which matrix will be used for every round (for the Algorithm). 1) and also don't know which matrices are used in which round of the encryption/decryption process (for Algorithm 2). Moreover, they don't know when we use which algorithm. The reason is that these algorithms and these matrices are chosen depending on the secret key that is not known by cryptanalysts.

On the other hand, since the matrix transformations preserve some MDS properties, we always have the matrices M_K or A_K, B_K as MDS ones. This makes adversaries who can know M_K or A_K, B_K, still face the largest branch number.

Suppose that the cryptanalysts found a lot of plaintext/ciphertext pairs to serve their attack. To predict the MDS matrices used in the rounds, they must rely on unknown keys or they must exhaustively search for the possible MDS matrices. Thus, with the obtained data, they will try with each of these MDS matrices for cryptanalysis. The result is that the computation complexity increases significantly (under the exhaustive method). Specifically:

Computation complexity for cryptanalysis of dynamic block cipher = (Computation complexity for cryptanalysis of static block cipher) × (the number of possible dynamic MDS matrices).

Furthermore, according to Shannon [15], the solution's unique interval of a cryptosystem is defined as the minimum number of ciphertext characters on which the cryptanalysts can uniquely find the secret key. In practice, the designers want the solution unique interval to be as large as possible to increase the security of the block cipher. The solution unique interval is defined by:

$$\text{Solution unique interval} = \frac{\text{Entropy(key)}}{language\ redundancy}$$

Thus, the larger the key space, the larger the solution unique interval.

For block ciphers that are made dynamically at the diffusion layer, since the diffusion layer or dynamic MDS matrices are hidden, there are two secret parameters: first is the secret key, and the other is the dynamic MDS matrices. Thus, it is clear that the dynamic cases increase the key space thanks to the dynamic MDS matrices, and

thus increase the security of the block cipher significantly.

For the two proposed algorithms, it can be seen that: Algorithm 1 is simpler than Algorithm 2, but in terms of security, Algorithm 2 has higher security than Algorithm 1, because in Algorithm 2 we use two MDS matrices instead of one matrix as in Algorithm 1. Moreover, these two matrices are used alternately through rounds, so the cryptanalysts have to find out one more thing to see which matrix is used in which round. Of course, the cryptanalysts do not have any basis for predicting this information. The ability to apply the above algorithms will depend on the real needs of users.

It can be seen that, with dynamic MDS matrices, the diffusion layer in block cipher becomes much more unpredictable, thus increasing the strength of the block cipher.

Table 1 provides a comparison between the proposed dynamic methods for the SPN block ciphers (Algorithm 1, Algorithm 2) with the dynamic methods proposed in [3] and [8].

TABLE 1: COMPARISON OF THE PROPOSED DYNAMIC ALGORITHMS WITH THE DYNAMIC METHODS IN [3, 8]

Method / **Criteria**	**Dynamic methods in [3, 8]**	**Proposed dynamic algorithms**
Dynamic by key	Yes	Yes
The number of original MDS matrices used	One	One or two
Matrix transformation used	Scalar multiplication, Permutation of row and column	Direct exponent, Scalar multiplication
Use a random bit generator function	Yes	No
The ability to preserve some good cryptographic properties of the original MDS matrices	No	Yes

V. CONCLUSION

In this paper, we propose algorithms to build a dynamic diffusion layer for SPN block ciphers based on the direct exponent and scalar multiplication transformation. These transformations are capable of preserving the good cryptographic properties of the MDS matrix when made dynamically. Therefore, the proposed dynamic algorithms contribute to improving the security of the SPN block ciphers against strong attacks on the block cipher such as a linear attack and differential attack.

REFERENCES

[1] G. Murtaza, N. Ikram, "Direct Exponent and Scalar Multiplication Classes of an MDS Matrix", [EB/OL], National University of Sciences and Technology, Pakistan, (2011-01-10), pp. 2-5.

[2] K.C Gupta, I.G Ray, "On Constructions of MDS Matrices From Circulant-Like Matrices For Lightweight Cryptography", Technical Report No. ASU/2014/1, Dated : 14th February, 2014.

[3] W. Mohamed, Ridza, M. Abdulrashid, "A method for linear transformation in substitution permutation network symmetric-key block cipher," international application published under the patent cooperation treaty, 10 may 2012, pp. 3-14.

[4]. T. T. Luong, N. N. Cuong, L. T. Dung, "The preservation of good cryptographic properties of MDS matrix under direct exponent transformation", Journal of Computer Science and Cybernetics, vol.31, no.4, pp. 291–303, 2015.

[5]. T. T. Luong, N. N. Cuong, L. T. Dung, "A new statement about direct exponent of an MDS matrix in block ciphers", in 2015 IEEE the Seventh International Conference on Knowledge and Systems Engineering (KSE), IEEE, pp. 340–343, 2015. (Date Added to IEEE Xplore: 07 January 2016).

[6]. T. T. Luong, N. N. Cuong, L. T. Dung, "The preservation of the coefficient of fixed points of an MDS matrix under direct exponent transformation", in 2015 IEEE International Conference on Advanced Technologies for Communications (ATC), IEEE, pp. 111–116, 2015. (Date Added to IEEE Xplore: 25 January 2016).

[7]. T. T. Luong, N. N. Cuong, "Direct exponent and scalar multiplication transformations of mds

matrices: some good cryptographic results for dynamic diffusion", Journal of Computer Science and Cybernetics, vol.32, no.1, pp. 1–17, 2016.

[8] G. Murtaza, A. A. Khan, S. W. Alam, A. Farooqi, "Fortification of aes with dynamic mix-column transformation," IACR Cryptology ePrint Archive, vol. 2011, p. 184, 2011.

[9] F. Ahmed and D. Elkamchouchi, "Strongest aes with s-boxes bank and dynamic key mds matrix (sdk-aes)," International Journal of Computer and Communication Engineering, vol. 2, no. 4, p. 530, 2013.

[10] F.J. MacWilliams, N.J.A. Sloane, The theory of error-correcting codes. Elsevier, 1977.

[11] M.R.Z'aba, Analysis of Linear Relationships in BlockCiphers. Ph.D. Thesis, Queensland University of Technology, Brisbane, Australia, 2010.

[12] Heys H.M. and Tavares S.E. (1996), "The design of product ciphers resistatnt to dierential and linear crypt-analysis", Journal of cryptography, vol. 9, no. 1, pp. 1-19.

[13] Lai X., Massey J.L. and Murphy S. (1991), "Markov ciphers and differential cryptanalysis", In Proceedings of Advances in Cryptology, LNCS 473, Springer, pp. 389 - 404.

[14] Matsui M. (1994), "Linear cryptanalysis method for des cipher", Advances in Cryptology|EUROCRYPT'93, LNCS 765, pp. 386-397, Springer-Verlag.

[15] Shannon C.E. (1949), "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656-715.

ABOUT THE AUTHOR

Tran Thi Luong

Workplace: Academy of Cryptography Techniques, No.141 Chien Thang road, Tan Trieu, Thanh Tri, Hanoi, Vietnam

Email: luongtran@actvn.edu.vn

Education: Bachelor of Mathematics and Informatics of The Ha Noi university of Sicence in 2006; Master degreee in cryptographic technique at Academy of Cryptographic Techniques in 2012; Phd degree in cryptographic technique at Academy of Cryptographic Techniques in 2019;

Recent research direction: Cryptogaphy, Coding theory and Information Security.