Journal of Computer Science and Cybernetics, V.38, N.4 (2022), 365–375
DOI no 10.15625/1813-9663/38/4/17226

FAST COMPUTATION OF DIRECT EXPONENTIATION TO SPEED UP IMPLEMENTATION OF DYNAMIC BLOCK CIPHERS

LUONG TRAN THI

Academy of Cryptography Techniques, 141 Chien Thang Street, Tan Trieu Ward, Thanh Tri District, Ha Noi, Viet Nam

Abstract. MDS (maximum distance separable) matrices are ones that come from MDS codes that have been studied for a long time in error correcting code theory and have many applications in block ciphers. To improve the security of block ciphers, dynamic block ciphers can be created. Using MDS matrix transformations is a method used to make block ciphers dynamic. Direct exponentiation is a transformation that can be used to generate dynamic MDS matrices to create a dynamic diffusion layer of the block ciphers. However, for cryptographic algorithms that use an MDS matrix as a component of them, the implementation of matrix multiplication is quite expensive, especially when the matrix has a large size. In this paper, the mathematical basis for quick calculation of direct exponentiation of an MDS matrix will be presented. On that basis, it is to suggest how to apply that fast calculation to dynamic algorithms using the direct exponentiation. This result is very meaningful in software implementation for MDS matrices, especially when implementing dynamic block ciphers to increase execution speed.

Keywords. MDS matrix, direct exponentiation, dynamic algorithms.

1. INTRODUCTION

Block ciphers are an important area of modern cryptography. The application of block ciphers to design cryptographic products for the national security field of socio-economics has many advantages and efficiency. In cryptography, a block cipher is a symmetric key algorithm that operates on fixed-length groups of bits, called blocks, with a stable transformation. The most commonly used block ciphers are the Feistel and SPN (Substitution Permutation Network) structures. The diffusion layer of SPN block ciphers is usually built on the basis of MDS matrices which are the matrices of the maximum distance separable codes (MDS codes).

To improve the security of block ciphers, one can generate dynamic block ciphers. In addition to the diffusion and substitution layers of the SPN block ciphers, the authors in [1,2] recently proposed a way to animate with new XOR tables, implemented at the key addition layer. Besides, many ways of animating block ciphers have been studied as in [3–8]. In [3], Al-Wattar et al. proposed a method to animate the AES block cipher at the diffusion layer. Specifically, instead of using a static MixColumn transformation, the approach is to use a dynamic MixColumn transformation based on key dependent DNA structures and processes.

Corresponding author.
E-mail addresses: luongtranhong@gmail.com

In [4], Ahmed et al. introduced a modified AES algorithm with a bank of potential sboxes that behave like a rotating block mechanism and a dynamic MDS matrix (this algorithm is called SDK-AES) that is, animate at both substitution and diffusion layers. In [5], Murtaza et al. proposed to replace the Mixcolumn in AES with a dynamic Mixcolumn. The dynamic MDS matrix is generated from AES's MDS matrix with scalar multiplication in rows and an additional key. In [6], Ismil et al. proposed a dynamic SPN block cipher based on AES (DRAES), where the animated transformation is a rotation whose amount of rotating depends on the data (plaintext and ciphertext) in AddRoundKey and on the key in the key extension of AES. In [7], Bai et al. proposed a dynamic block cipher algorithm with variable size, designed with unlimited size of keys, the permutation changing dynamically based on the encryption key, and the block size changing in each round. In [8], Mohamed et al. proposed a method to construct a dynamic diffusion layer for SPN block ciphers. This method generates key-dependent MDS matrices from a given $m \times m$ MDS matrix, by scalar multiplication and permutations of elements in the given matrix.

Thus, it is possible to see a variety of methods for animating SPN block ciphers to improve the security of these block ciphers against many strong attacks. To animate the SPN block cipher, in some cases it is common to use MDS matrix transformations such as: direct exponentiation, scalar multiplication, and permutations of rows or columns of an MDS matrix.

The direct exponentiation was first introduced by Ghulam Murtaza and Nassar Ikram in [9] but the authors have not shown the ability to preserve good cryptographic properties of this transformation. In [9], the authors did not show how to compute direct exponentiation effectively. In [10–12], we showed the ability to preserve many good cryptographic properties of the direct exponentiation. In [13], we proposed two dynamic diffusion algorithms to generate dynamic SPN block ciphers based on direct exponentiation and scalar multiplication. This preservation ability is very important to ensure that dynamic block ciphers are still much more secure than static block ciphers. However, in [13], we did not show the computational and practical performance of the direct exponentiation.

For SPN block ciphers, implementing MDS matrix multiplication at the diffusion layer is quite expensive, especially when the matrix has a large size. In this paper, we will present a quick calculation method of direct exponentiation to apply to dynamic block cipher algorithms. This result is very meaningful in software implementation when implementing dynamic block ciphers to increase execution speed.

The paper is organized as follows. In Section 2, preliminaries about MDS matrices and the direct exponentiation are introduced. Section 3 presents the mathematical basis for the quick calculation of direct exponentiation and the ability to apply this fast calculation to dynamic algorithms using direct exponentiation. The conclusions of the paper are presented in Section 4.

2. PRELIMINARIES AND RELATED WORKS

2.1. MDS matrix

MDS matrices are ones that come from MDS codes in error correction code theory. A code $[n, k, d]$ is one from error correcting code theory of length n, number of dimensions k, and a minimum distance of d.

In [14] there is an important theorem about the MDS matrix as follows.

Theorem 1 ([14], page 321). *A code $[n, k, d]$ with a generator matrix $G = [I|A]$ where A is a $k\times(n-k)$ matrix and I is an identity matrix of order, k is MDS if and only if every square submatrix (generated from any i rows and any i columns, for any $i = 1, 2, \dots,$ $\min\{k, n-k\}$ of A is nonsingular.*

From Theorem 1, an MDS matrix can be defined that is a matrix whose all square sub-matrices is nonsingular.

2.2. Direct exponentiation of an MDS matrix

Ghulam Murtaza and Nassar Ikram were the first authors that introduced the definition of the direct exponentiation of an MDS matrix [9]. The concept of direct exponential matrix is also given by the authors as follows.

Definition 1 [9]. Let F be a Galois field. Let matrix $A = [a_{i,j}]_{m\times m}$, $a_{i,j} \in F$, then $A_{d^e} = \left[a_{i,j}^e\right]_{m\times m}$, $(e = 1, 2, 3\dots)$ is called the direct e exponent matrix of A. And A_{d^2} is called direct square matrix of A.

In [10–12], we showed that direct exponentiation is able to preserve many good cryptographic properties of MDS matrices such as MDS property, recursive, involutory, circulant and circulant-like, number of fixed points and coefficient of fixed points, etc. This conservation of the direct exponential transformation is very useful for application to dynamic block ciphers.

3. SOME RESULTS

In this section, we will show how to quickly calculate direct exponentiation over the field $GF(p^r)$, p is prime. Galois fields with p^r elements $GF(p^r)$ can be built as an extension of the field $GF(p)$, being r any integer greater than 1. The Galois field $GF(p^r)$ can be defined using polynomials with coefficients belonging to $GF(p)$ and degrees smaller than r, and the number of elements of this field is p^r. From there, this fast calculation can be applied to dynamic algorithms to increase the execution speed in the software. That is, in the software implementation of dynamic block cipher algorithms that are performed at the diffusion layer, instead of directly calculating the dynamic MDS matrix from an original matrix by direct exponentiation of each element in the matrix, we can use the lookup table suggested in this section to compute the dynamic MDS matrix that is much faster than calculating for each element.

3.1. Mathematical basis for quick calculation of the direct exponentiation

One more point to note about the usefulness of direct exponentiation is that the power of 2^k of each element in a matrix is performed very quickly in software for elements over $GF(2^r)$. Indeed, suppose A is a matrix over $GF(2^r)$ with the generator polynomial of this field denoted by $p(x)$ of degree r, and consider any element a in A. Note that, the generator polynomial of the is an irreducible polynomial that the results of every polynomial addition and multiplication in the field must be modulo for this polynomial.

Then, the element $a \in GF\left(2^r\right)$ will have the polynomial representation as follows

$$a = a_0 + a_1 x + a_2 x^2 + \ldots + a_{r-1} x^{r-1},$$

where, $a_i \in GF\left(2\right)$, $0 \leq i \leq r-1$.

When performing direct 2^k exponent of the matrix A, each element in the matrix will be increased to an exponent of 2^k.

It is to have $a^{2^k} = \left(a_0 + a_1 x + a_2 x^2 + \ldots + a_{r-1} x^{r-1}\right)^{2^k}$. We already know that, if elements $a_i \in GF\left(2^r\right)$, $(i = 1, 2, \ldots n)$

$$(a_1 + a_2 + \ldots + a_n)^2 = (a_1^2 + a_2^2 + \ldots + a_n^2). \tag{1}$$

Applying (1), it is to have

$$a^{2^k} = a_0^{2^k} + a_1^{2^k} x^{2^k} + a_2^{2^k} x^{2^{k+1}} + \ldots + a_{r-1}^{2^k} x^{(r-1)2^k}.$$

Since $a_i \in GF\left(2\right)$, it should always be $a_i^{2^k} = a_i$, $(0 \leq i \leq r-1)$ so no further multiplication is required to calculate these coefficients. Hence, it is to get

$$a^{2^k} = a_0 + a_1 x^{2^k} + a_2 x^{2^{k+1}} + \ldots + a_{r-1} x^{(r-1)2^k}.$$

Then get modulo this polynomial by $p\left(x\right)$, it is to get an element $a^{2^k} \in GF\left(2^r\right)$.

In the following, we will consider how to quickly calculate direct exponent over $GF\left(2^r\right)$.

First, consider division modulo $a^2 = a_0 + a_1 x^2 + a_2 x^4 + \ldots + a_{r-1} x^{2(r-1)}$ by $p\left(x\right)$.

Let $m = \frac{r}{2}$, it is to get

$$a^2 = a_0 + a_1 x^2 + a_2 x^4 + \ldots + a_m x^{2m} + \ldots + a_{r-1} x^{2(r-1)}. \tag{2}$$

The items x^{2i} $(0 \leq i < m)$ do not need to consider because their degrees is less than r, so no need to get division modulo by $p\left(x\right)$.

Considering the items x^{2i} $(m \leq i \leq 2\left(r-1\right))$, these are items having degree greater than r and therefore they are needed to be divided modulo by $p\left(x\right)$. The results of these divisions are prepared and included in Table 1.

Table 1: Result of division modulo a^2 by $p\left(x\right)$

i	x^{2i}	$x^{2i} \mod p\left(x\right)$
m	x^{2m}	$c_m\left(x\right) = c_{m_0} + c_{m_1} x + c_{m_2} x^2 + \ldots + c_{m_{r-1}} x^{r-1}$
$m+1$	$x^{2(m+1)}$	$c_{m+1}\left(x\right) = c_{(m+1)_0} + c_{(m+1)_1} x + c_{(m+1)_2} x^2 + \ldots + c_{(m+1)_{(r-1)}} x^{r-1}$
$\ldots$	$\ldots$	$\ldots$
$r-1$	$x^{2(r-1)}$	$c_{r-1}\left(x\right) = c_{(r-1)_0} + c_{(r-1)_1} x + c_{(r-1)_2} x^2 + \ldots + c_{(r-1)_{(r-1)}} x^{r-1}$

From (2) and the results of the Table 1, it is to have

$$\begin{aligned}
&a^2 \mod p(x)\\
&= \Big(a_0 + a_1x^2 + a_2x^4 + \ldots + a_{m-1}x^{2(m-1)}\Big) + \Big(a_mx^{2m} + \ldots + a_{r-1}x^{2(r-1)}\Big) \mod p(x)\\
&= \Big(a_0 + a_1x^2 + a_2x^4 + \ldots + a_{m-1}x^{2(m-1)}\Big) + \Big(a_mx^{2m} + \ldots + a_{r-1}x^{2(r-1)}\Big) + \sum_{i=m}^{r-1} c_i(x)\\
&= \Big(a_0 + a_1x^2 + a_2x^4 + \ldots + a_{m-1}x^{2(m-1)}\Big) + (\sum_{i=m}^{r-1} c_{i_0} + \sum_{i=m}^{r-1} c_{i_1}x + \ldots + \sum_{i=m}^{r-1} c_{i_{(r-1)}}x^{(r-1)})\\
&= (a_0 + \sum_{i=m}^{r-1} c_{i_0}) + \sum_{i=m}^{r-1} c_{i_1}x + (a_1 + \sum_{i=m}^{r-1} c_{i_2})x^2 + \ldots + (a_{m-1} + \sum_{i=m}^{r-1} c_{i_{(m-1)}})x^{m-1} + \sum_{i=m}^{r-1} c_{i_m}x^m\\
&\quad + \ldots + \sum_{i=m}^{r-1} c_{i_{(r-1)}}x^{(r-1)}. \qquad (3)
\end{aligned}$$

Let
$$a^2 = \quad a_{1_0} + a_{1_1}x + a_{1_2}x^2 + \ldots + a_{1_{(r-1)}}x^{r-1}. \qquad (4)$$

From (3) and (4), deduce

$$\begin{cases}
a_{1_0} = a_0 + \sum\limits_{i=m}^{r-1} c_{i_0}\\
a_{1_1} = \sum\limits_{i=m}^{r-1} c_{i_1}\\
a_{1_2} = a_1 + \sum\limits_{i=m}^{r-1} c_{i_2}\\
\vdots\\
a_{r_1} = \sum\limits_{i=m}^{r-1} c_{i_{r-1}}.
\end{cases} \qquad (5)$$

Thus, for any element $a \in GF(2^r)$, we can calculate division $a^2 \mod p(x)$ very quickly by using the built-in table (Table 1), and the result of this division is calculated by the formulas (3) and (5).

Next, consider division modulo $a^{2^2} \mod p(x)$

It is to have $a^{2^2} \mod p(x) = \left(a^2\right)^2 \mod p(x)$.

Then, the role of a^2 in this formula is the same as that of a in (3).

From Table 1, formulas (3) and (4), it is to have

$$\begin{aligned}
a^{2^2} \mod p(x) = (a_{1_0} + \sum_{i=m}^{r-1} c_{i_0}) + \sum_{i=m}^{r-1} c_{i_1}x + (a_{1_1} + \sum_{i=m}^{r-1} c_{i_2})x^2 + \ldots +\\
(a_{1_{(m-1)}} + \sum_{i=m}^{r-1} c_{i_{(m-1)}})x^{m-1} + \sum_{i=m}^{r-1} c_{i_m}x^m + \ldots + \sum_{i=m}^{r-1} c_{i_{(r-1)}}x^{(r-1)}. \qquad (6)
\end{aligned}$$

Let
$$a^{2^2} = a_{2_0} + a_{2_1}x + a_{2_2}x^2 + \ldots + a_{2_{(r-1)}}x^{r-1}. \qquad (7)$$

From (6), deduce

$$\begin{cases} a_{2_0} = a_{1_0} + \sum\limits_{i=m}^{r-1} c_{i_0} \\ a_{2_1} = \sum\limits_{i=m}^{r-1} c_{i_1} \\ a_{2_2} = a_{1_1} + \sum\limits_{i=m}^{r-1} c_{i_2} \\ \vdots \\ a_{2_{(r-1)}} = \sum\limits_{i=m}^{r-1} c_{i_{r-1}}. \end{cases} \tag{8}$$

By induction, it is to get

$$a^{2^k} = \quad a_{k_0} + a_{k_1}x + a_{k_2}x^2 + \ldots + a_{k_{(r-1)}}x^{r-1}, \tag{9}$$

where

$$\begin{cases} a_{k_0} = a_{(k-1)_0} + \sum\limits_{i=m}^{r-1} c_{i_0} \\ a_{k_1} = \sum\limits_{i=m}^{r-1} c_{i_1} \\ a_{k_2} = a_{1_1} + \sum\limits_{i=m}^{r-1} c_{i_2} \\ \vdots \\ a_{k_{(r-1)}} = \sum\limits_{i=m}^{r-1} c_{i_{r-1}}, \end{cases} \tag{10}$$

for $(1 \le k \le r-1)$.

3.2. Applying fast calculation to dynamic algorithms that use direct exponentiation

Later, in some dynamic algorithm that is made dynamic in the diffusion layer, it is necessary to perform direct exponentiation of a given MDS matrix A over $GF(2^r)$. We will apply the above fast calculation to achieve speed efficiency when creating dynamic MDS matrices.

Suppose the matrix $A = [a_{i,j}]_{m\times m}$, $a_{i,j} \in GF(2^r)$ has c distinct elements that are all other than 1, denoted as $a_1, a_2, \ldots, a_c$. (Since the element 1 does not change by the direct exponentiation, we consider only the elements other than 1 of A.)

Suppose in a dynamic algorithm that requires calculating the direct 2^{t_0} exponent of matrix A to generate a new MDS matrix $A_{d^{2^{t_0}}}$, where t_0 is a selected number satisfying $1 \le t_0 \le r-1$.

Using the quick calculation of direct exponentiation presented in the previous section, we can make Table 2 to obtain the elements $(a_i)^{2^{t_0}}$ $(1 \le i \le c)$.

In Table 2, a_{i_j} (for $1 \le i \le t_0$, $1 \le j \le c$) are elements over $GF(2^r)$, which can be presented in the Hexa form.

Table 2: Lookup table of direct 2^k exponent for $1 \le k \le t_0$

Element $a_i \in A$	$(a_i)^{2^k}$			
	$k=1$	$k=2$	$\dots$	$k=t_0$
a_1	a_{1_1}	a_{2_1}	$\dots$	$a_{t_0 1}$
a_2	a_{1_2}	a_{2_2}	$\dots$	$a_{t_0 2}$
$\dots$	$\dots$	$\dots$	$\dots$	
a_c.	a_{1_c}	a_{2_c}	$\dots$	$a_{t_0 c}$

Based on the Table 2, it is easy to calculate the direct exponentiation matrix $A_{d^{2^{t_0}}}$ of the matrix A.

More general, we consider the two following cases.

Consider the finite field is $GF(2^r)$

In order to be able to compute easily the direct 2^k exponentiation matrices of any MDS matrix A with the elements over $GF(2^r)$, we can prepare a table of size $(2^r-1)\times(r-1)$ (if necessary) includes all elements that are direct 2^k exponentiation (for $1 \le k \le r-1$) of all elements other than 0 and 1 of the field $GF(2^r)$. Note that direct exponentiation is still calculated fastly as showed in Section 3.1. This table has the following form, where the elements are represented in the Hexa form.

Table 3: Lookup table of direct 2^k exponent (for $1 \le k \le t_0$) over $GF(2^r)$

Elements $\alpha_i \in GF(2^r)$	$(a_i)^{2^k}$			
	$k=1$	$k=2$	$\dots$	$k=r-1$
α_1	a_{1_1}	a_{2_1}	$\dots$	$a_{(r-1)_1}$
α_2	a_{1_2}	a_{2_2}	$\dots$	$a_{(r-1)_2}$
$\vdots$	$\dots$	$\dots$	$\dots$	$\dots$
$\alpha_{2^{r-2}}$	$a_{1_{2^{r-2}}}$	$a_{2_{2^{r-2}}}$	$\dots$	$a_{(r-1)_{2^{r-2}}}$

where a_{i_j} (with $1 \le i \le r-1$, $1 \le j \le 2^{r-2}$) are elements over $GF(2^r)$ that can be represented in the Hexa form.

Thus, direct exponentiation of MDS matrices over $GF(2^r)$ will be performed very quickly in software. We know that most of the cryptographic algorithms today use the base field is $GF(2^r)$, so the direct exponentiation will be very convenient in software implementation when the cryptographic algorithms use it.

Consider the general field is $GF(p^r)$, for p is a prime number

The fast calculation in this section is considered when looking up the table (not counting the time to create the lookup table) during the encryption and decryption process.

Denote the non-zero elements of the field $GF(p^r)$ by $\{b_1, b_2, \dots, b_{p^r-1}\}$.

Denote $|b_i|$ is the corresponding decimal value of b_i $(1 \le i \le p^r-1)$. For example, if b_i has a vector representation as $b_i = (t_0, t_1, \dots, t_{r-1})$ then $|b_i| = \sum_{i=0}^{r-1} t_i p^i$.

Let X be a one-dimensional array consisting of p^r-1 elements $X[1], X[2], \dots, X[p^r-1]$. The elements in the array X receive values that are powers of p of p^r-1 non-zero elements in the field $GF(p^r)$. For simplicity, denote that the field element b_i has its corresponding decimal value equal to i for $1 \le i \le p^r-1$, then we have $X[i] = (b_i)^p$.

Assuming we need calculate $(b_i)^{p^k}$, we perform the above array lookup as follows.

Since $X[i] = b_i^p$ then $X[|b_i^p|] = (b_i^p)^p = b_i^{p^2}$, and $X\left[\left|b_i^{p^2}\right|\right] = \left(b_i^{p^2}\right)^p = b_i^{p^3}$, and so on.

Continuing like this we have $X\left[\left|b_i^{p^{k-1}}\right|\right] = \left(b_i^{p^{k-1}}\right)^p = b_i^{p^k}$.

Look-up table creation. For the field $GF(p^r)$, construct a table of size $(p^r-1)\times r$, consisting of p^r-1 rows and r columns. The table entries will store the values of the powers $p^k\,(1\le k\le r)$ of all the elements in $GF(p^r)$. The i-th row corresponds to the field element b_i, $1\le i\le p^r-1$. The entry in row i, column k of the table is the power p^k $(1\le k\le r)$ of the field element b_i, that is: $(b_i)^{p^k}$.

To create this table, first we only need to calculate column 1 (corresponding to $k=1$), the remaining columns are calculated by looking up the table once, for example, column 2 (corresponding to $k=2$) only needs to look up the table once. times through column 1, column 3 looks up the table through column 2, and so on.

Application. For any MDS matrix $A = [a_{i,j}]_{m\times m}, a_{i,j}\in GF(p^r)$, to calculate the direct $p^k\,(1\le k\le r)$ exponential matrix of the matrix A in the usual way, we have to compute p^k powers of m^2 elements in the matrix A. But when using the pre-created lookup table as above, we only need to look up the table for m^2 elements in A. Table lookup operation is very fast, just O(1) *complexity.* Through this table, it is possible to quickly and easily find the direct p^k exponent matrix of any MDS matrix. Moreover, this lookup table can be used forever for many different applications that use the direct exponential matrices of an MDS matrix, so it really has a very good application in practice to improve speed.

Example 1. Consider the field $GF(2^4)$ with the primitive polynomial is x^4+x+1.

Denote that an element $b_i\ne 0$ has corresponding decimal value i for $1\le i\le 15$. So, 15 non-zero elements of the above $GF(2^4)$ include

$\{b_1=1,\ b_2=x,\ b_3=x+1,\ b_4=x^2,\ b_5=x^2+1,\ b_6=x^2+x,\ b_7=x^2+x+1,$
$b_8=x^3,\ b_9=x^3+1,\ b_{10}=x^3+x,\ b_{11}=x^3+x+1,\ b_{12}=x^3+x^2,\ b_{13}=x^3+x^2+1,$
$b_{14}=x^3+x^2+x,\ b_{15}=x^3+x^2+x+1\}.$

Construct a lookup table of size 15×4 (Table 4), the i-th row is the 2^k powers of the element b_i for $1\le k\le 4$ and $1\le i\le 15$. The element in row i, column k of the table is: $(b_i)^{2^k}$. We only need to compute the first column in the Table 4, that is, the values $X[i]=b_i^2$ $(1\le i\le 15)$. The values of the remaining entries do not need to be calculated, but only need to look up the table based on the previous column.

For example, consider row number 6, it is to have $(b_6)^{2^1} = X[6] = b_{13}$. Then, look up Table 1:

+ To calculate $(b_6)^{2^2}$, because the first column (corresponding to $k=1$) is equal to b_{13}, just look at row 13, column 1 infer: $(b_6)^{2^2} = b_7$ (because $(b_6)^{2^2} = (b_{13})^{2^1}$).
+ To calculate $(b_6)^{2^3}$, because the second column is equal to b_7, just look at row 7, column 1 to infer: $(b_6)^{2^3} = b_{12}$.
+ To calculate $(b_6)^{2^4}$, because the third column is equal to b_{12}, so just look at row 12, column 1 to infer: $(b_6)^{2^4} = b_6$.

Example 2. Consider the field $GF(3^3)$ with the primitive polynomial is $2x^3+x^2+2$.

Table 4: Direct exponential lookup table over $GF\left(2^4\right)$

	$k=1$	$k=2$	$k=3$	$k=4$
1	$X[1]=b_1$	b_1	b_1	b_1
2	$X[2]=b_4$	b_9	b_{14}	b_2
3	$X[3]=b_5$	b_8	b_{15}	b_3
4	$X[4]=b_9$	b_{14}	b_2	b_4
5	$X[5]=b_8$	b_{15}	b_3	b_5
6	$X[6]=b_{13}$	b_7	b_{12}	b_6
7	$X[7]=b_{12}$	b_6	b_{13}	b_7
8	$X[8]=b_{15}$	b_3	b_5	b_8
9	$X[9]=b_{14}$	b_2	b_4	b_9
10	$X[10]=b_{11}$	b_{10}	b_{11}	b_{10}
11	$X[11]=b_{10}$	b_{11}	b_{10}	b_{11}
12	$X[12]=b_6$	b_{13}	b_7	b_{12}
13	$X[13]=b_7$	b_{12}	b_6	b_{13}
14	$X[14]=b_2$	b_4	b_9	b_{14}
15	$X[15]=b_3$	b_5	b_8	b_{15}

Denote that an element $b_i \neq 0$ has corresponding decimal value i for with $1 \leq i \leq 26$. Then 26 non-zero elements of the above $GF\left(3^3\right)$ include

$$\{b_1 = 1,\ b_2 = 2,\ b_3 = x,\ b_4 = x+1,\ b_5 = x+2,\ b_6 = 2x,\ b_7 = 2x+1,$$
$$b_8 = 2x+2,\ b_9 = x^2,\ b_{10} = x^2+1,\ b_{11} = x^2+2,\ b_{12} = x^2+x,$$
$$b_{13} = x^2+x+1,\ b_{14} = x^2+x+2,\ b_{15} = x^2+2x,\ b_{16} = x^2+2x+1,$$
$$b_{17} = x^2+2x+2,\ b_{18} = 2x^2,\ b_{19} = 2x^2+1,\ b_{20} = 2x^2+2,$$
$$b_{21} = 2x^2+x,\ b_{22} = 2x^2+x+1,\ b_{23} = 2x^2+x+2,\ b_{24} = 2x^2+2x,$$
$$b_{25} = 2x^2+2x+1,\ b_{26} = 2x^2+2x+2\}.$$

Construct a lookup table of size 26×3 (Table 5), the i-th row is the 3^k powers of the element a_i for $1 \leq k \leq 3$ and $1 \leq i \leq 26$. The element in row i, column k of the table is: $(b_i)^{3^k}$. Similar to Example 1, we only need to compute the first column in the Table 5, that is, the values $X[i] = b_i^3$ $(1 \leq i \leq 26)$. The values of the remaining entries do not need to be calculated, but only need to look up the table based on the previous column.

The way to look up Table 5 is similar to Example 1.

Remark. With such lookup tables, making these tables is very simple because we only need to calculate column 1, the remaining columns just need to look up the table as above. After having these lookup tables that store values of powers of non-zero elements in the field $GF(p^r)$, applications or programs that implement dynamic cryptographic algorithms using direct exponentiation need only look up the tables to get these values directly without having to recalculate, thus increasing execution speed. For example, with an MDS matrix of size 8, with 64 elements, instead of having to calculate the power of these 64 elements and then modulo by the primitive polynomial of the field $GF(p^r)$, we just need to look up the table 64 times, the table lookup operation is very fast, equivalent to O(1) complexity. It can be seen that this fast calculation makes more sense when the GF finite field has a large number of elements, which can save considerable costs.

Table 5: Direct exponential lookup table over $GF\left(3^3\right)$

	$k=1$	$k=2$	$k=3$
1	$X\left[1\right]=b_1$	b_1	b_1
2	$X\left[2\right]=b_2$	b_2	b_2
3	$X\left[3\right]=b_{11}$	b_{26}	b_3
4	$X\left[4\right]=b_9$	b_{24}	b_4
5	$X\left[5\right]=b_{10}$	b_{25}	b_5
6	$X\left[6\right]=b_{19}$	b_{13}	b_6
7	$X\left[7\right]=b_{20}$	b_{14}	b_7
8	$X\left[8\right]=b_{18}$	b_{12}	b_8
9	$X\left[9\right]=b_{24}$	b_4	b_9
10	$X\left[10\right]=b_{25}$	b_5	b_{10}
11	$X\left[11\right]=b_{26}$	b_3	b_{11}
12	$X\left[12\right]=b_8$	b_{18}	b_{12}
13	$X\left[13\right]=b_6$	b_{19}	b_{13}
14	$X\left[14\right]=b_7$	b_{20}	b_{14}
15	$X\left[15\right]=b_{16}$	b_{17}	b_{15}
16	$X\left[16\right]=b_{17}$	b_{15}	b_{16}
17	$X\left[17\right]=b_{15}$	b_{16}	b_{17}
18	$X\left[18\right]=b_{12}$	b_8	b_{18}
19	$X\left[19\right]=b_{13}$	b_6	b_{19}
20	$X\left[20\right]=b_{14}$	b_7	b_{20}
21	$X\left[21\right]=b_{23}$	b_{22}	b_{21}
22	$X\left[22\right]=b_{21}$	b_{23}	b_{22}
23	$X\left[23\right]=b_{22}$	b_{21}	b_{23}
24	$X\left[24\right]=b_4$	b_9	b_{24}
25	$X\left[25\right]=b_5$	b_{10}	b_{25}
26	$X\left[26\right]=b_3$	b_{11}	b_{26}

4. CONCLUSION

MDS matrix plays an important role in improving the security of block ciphers. However, implementing these matrices is quite expensive, especially when they are large in size. In this paper, the mathematical basis for how to quickly calculate direct exponentiation of MDS matrices has been presented, thereby oviding a thod to apply this fast calculation to dynamic algorithms using direct exponentiation. When there are built-in tables for values of direct exponentiation of non-zero elements in the field $GF\left(p^r\right)$, applications or programs that implement cryptographic algorithms need only look up the table to get these values directly without having to recalculate, thus increasing execution speed. It can be seen that this fast calculation makes more sense when the GF. finite field has a large number of elements, which can save significant costs. These results are of great practical significance because there is a possibility to increase the performance when implementing MDS matrices, especially when implementing dynamic block ciphers that use direct exponentiation.

REFERENCES

[1] A. I. Salih, A. Alabaichi, and A. S. Abbas, "A novel approach for enhancing security of advance encryption standard using private xor table and 3d chaotic regarding to software quality factor," *ICIC Express Letters Part B: Applications ICIC International*, vol. 10, no. 9, pp. 823–832, 2019.

[2] A. I. Salih, A. Alabaichi, and A. Y. Tuama, "Enhancing advance encryption standard security

based on dual dynamic xor table and mixcolumns transformation," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 3, pp. 1574–1581, September 2020.

[3] A. H. Al-Wattar, R. Mahmod, Z. A. Zukarnain, and N. Udzir, "A new dna based approach of generating key dependent mixcolumns transformation," *International Journal of Computer Networks & Communications (IJCNC)*, vol. 7, no. 2, March 2015.

[4] 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.

[5] G. Murtaza, A. A. Khan, S. W. Alam, and A. Farooqi, "Fortification of aes with dynamic mix-column transformation," *IACR Cryptology ePrint Archive*, 2011, cryptology ePrint Archive, Paper 2011/184. [Online]. Available: https://eprint.iacr.org/2011/184

[6] I. A. Ismil, H. G. Galal, S. Khattab, and M. A. E. I. M. E. Bahtity, "Performance examination of aes encryption algorithm with constant and dynamic rotation," *International Journal of Reviews in Computing*, vol. 12, 2012.

[7] K. C. S. Bai, M. V. Satyanarayana, and P. A. Vijaya, "Variable size block encryption using dynamic-key mechanism (vbedm)," *International Journal of Computer Applications*, vol. 27, no. 7, August 2011.

[8] R. W. Mohamed and M. Abdulrashid, "A method for linear transformation in substitution permutation network symmetric-key block cipher," International Application Published Under the Patent Cooperation Treaty, May 2012, pp. 3–14.

[9] G. Murtaza and N. Ikram, "Direct exponent and scalar multiplication classes of an mds matrix," *Cryptology ePrint Archive*, 2011, paper 2011/151. [Online]. Available: https://eprint.iacr.org/2011/151

[10] T. T. Luong and 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.

[11] T. T. Luong, N. N. Cuong, and 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.

[12] ——, "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, 2015, pp. 111–116.

[13] T. T. Luong, "Building the dynamic diffusion layer for spn block ciphers based on direct exponent and scalar multiplication," *Journal of Science and Technology on Information Security of Viet Nam Government Information Security Commission*, vol. 1, no. 15, pp. 10–16, 2022.

[14] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* North-Holland Publishing Company, 1977.

Received on June 15, 2022
Accepted on December 01, 2022