Journal of Theoretical and Applied Information Technology
15th April 2021. Vol.99. No 7

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPLEMENTATION OF TEXT DATA SECURITY USING MODULAR MULTIPLICATION BASED BLOCK CIPHER MODIFICATION

[1]H KHAIR, [2]MARISCHA ELVENY*, [3]A M H PARDEDE, [4]S RAMADANI, [5]ACHMAD FAUZI

[1,3,4,5]STMIK Kaputama, Jl. Veteran No. 4A-9A, Binjai- Sumatera Utara, Indonesia

[2]Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

E mail: [1]husnul.khair@gmail.com, [2]marischaelveny@usu.ac.id*, [3]akimmhp@live.com, [4]suci.ramadani23@gmail.com, [5]fauzyrivai88@gmail.com

ABSTRACT

One implementation of the use of Information and Communication Technology (ICT) via the internet is sending text messages via email, social media, or other means of communication. Nearly all messages sent privately over a private network are secret messages that you don't want to share with other people, be they text messages, voice messages, picture messages, and video messages. But in reality, confidential information is often leaked so that confidential information can be spread to various irresponsible parties. The efforts made by these parties are aimed at misusing the data sent by destroying, intercepting, changing the original message according to the wishes of other parties for their own interests. This action can make confidential information or messages visible to people who are not responsible. In overcoming this data security issue, steps are needed to improve data security. One technique that can be used to secure data is to use the Modular Multiplication based Block Chipper (MMB) algorithm, which is a simple method that is not too complicated but is hidden so that the message to be sent is safe. In this research, we propose an algorithm that simplifies the use of MMB by directly multiplying each plaintext to the key, and changing the constant C using RoL so that the binary in constant C is doubled. The results of the implementation of this study concluded that the encryption of the original message would be longer, and it would be more difficult for other parties to crack.

Keywords: *Cryptography, MMB, Security, Text.*

1. INTRODUCTION

Information and Communication Technology is developing rapidly so that access to information obtained is very broad. One of the dissemination of information can be done through sending messages, messages that can be sent through social media, email, short messages or other media. The messages conveyed are sometimes in the form of messages that are confidential in nature so that not all parties can see the message. However, along with this development, there are also breaches or misuse in the security of the data sent, such as by destroying, tapping, changing the message for personal gain. This action can make information or messages that are confidential can be seen by people who are not responsible. Therefore, the issue of information security is very important in an information system for mutual security and personal security. For that, we need a security system that can protect information [1].

Data security technology continues to evolve from data encryption to data insertion. A new method of modular multiplication based on Karatsuba-like multiplication has been carried out by Zhen Gu et al. The method is useful for both special modulus such as NIST prime and general modulus based on Montgomery's modular multiplication, which results in an intermediate step between the multiplication of integers required to be simplified as one simple step [2].

Multiplication MODULAR and modular exponent in the implementation of securing text or data is widely used, operation in many public-key cryptosystems (PKCs) with large modulus is done by repeating modular multiplication [3], [4], which is a slow and very time-consuming operation [5], as a result of the slow and very slow operation [6]. this time consuming, so the dependence on the output rate of the modular multiplication and the number of modular multiplications required led to the performance of many PKC. A high output rate for large integers is difficult to obtain without using

supported hardware acceleration. Montgomery's modular multiplication is an efficient and highly recommended method of performing high-level hardware implementations of large modulus modular multiplication [7], [8], [9][10], [11]. This algorithm replaces experimental division with a series of addition and right shift operations [12], [13]. A very challenging follow-up study in Montgomery's modular multiplication was the time-consuming carry propagation of the addition of very large operands [14], [15].

Hardware implementation of the multibit-scan-multibit-shift technique using multiplier expansion and MBS (limited number of shifts), as well as the proposed modular multiplication architecture using the modified L2R and R2L modular exponential architectures. The results of the complexity analysis and implementation results show that the proposed architecture provides a significant increase in total computation time consumption which is faster and throughput is faster than other modular multiplication architectures [16].

Research conducted by Saldamli and Koc proposed an algorithm to for Montgomery Modular Multiplication (MMM) in the spectral domain [17]. However, their proposed modular spectral algorithm is derived from the serial-digit variant MMM [18], the proposed algorithm is essentially sequential, making it unfriendly for massively parallel computing. McLaughlin, et al proposed a new framework for a modified version of MMM which is suitable to be performed in the spectral domain [19]. The new MMM version has a lower multiplication time than the original MMM. To avoid doubling the length of the transformation at the time of multiplication, a cyclic convolution and a negative convolution are used. When multiplying MMM in the spectral domain, the FFT-based Montgomery Product Reduction (FMPR) Algorithm is used which is very suitable for use in parallel hardware designs [20],[21].

In previous studies by Bos and Friedberger, as well as Karmakar, et al., It was found that Montgomery's reduction of specific structural primes used in isogeny-based cryptography was not optimal [22], [23], [24], in that study it was found that a special modulus could be used. for the most effective implementation. faster. In the Diffie-Hellman (SIDH) supersingular isogeny, the prime number, p, is of the form, $p = f * 2^a3^b - 1$, where f is a minor number. Public key cryptographic systems are very similar to others [25], Karmakar et al. proposes an Efficient Finite Field Multiplication (EFFM) algorithm, where the main plane has a special structure that proposes two new algorithms, namely the Enhanced EFFM (Efficient Finite Field Multiplication) to FFM1 algorithm, which is then upgraded from the original EFFM by reducing the number of operands, and a very second algorithm FFM2. differs from the original EFFM and FFM1, and allows for a larger operand size while reducing the number of its modulus multiplication operations. From the two proposed algorithms, it is concluded that it can significantly speed up computation, and produce a better hardware architecture when compared to the original EFFM algorithm implementation [26].

In the development of the FFM1 and FFM2 algorithms, a mathematical transformation is applied to reduce the number of operations in the first new algorithm (FFM1), and better and faster results are obtained in the second FFM2 algorithm, which is 6 times faster than before. The hardware implementation of the FFM2 algorithm when compared to FFM1 and EFFM is the fastest. In addition, the FFM2 algorithm can be applied to various modulo, with limitations on the EFFM algorithm and the FFM1 algorithm. Also the FFM2 hardware implemented is 31% faster than the software implementation [27].

One technique that can be used to secure data is to use the Modular Multiplication based Block Chipper (MMB) algorithm, a simple method that is not too complex but the hidden message is quite safe [28]. The main advantages of a given cipher are ease of implementation and the possibility of probabilistic encryption. This means that text encryption will do it differently when the keys are the same and the data are the same. So, the encryption strength is increased. In addition, the size of the encoded message is difficult to predict [29].

The data security issue is the most important issue today. Many cryptosystems are offered to secure data, cryptosystems using public keys are the cryptographic systems most commonly used to secure data communications, one of which is by using modular multiplication which is the basic operation of common public-key cryptography systems such as RSA, ElGamal, and Elliptic Curve Cryptography. (ECC), Diffie-Hellman (DH) key agreement [30]. Cryptographic systems use a public key that is implemented as a coprocessor so that it can speed up encryption operations and reduce computation time or computational overhead of the main processor [31], [32].

The solution to the problem in this study is how to apply the Modular Multiplication Based Block

Cipher (MMB) method to protect data security in secret text messages, so that the text files cannot be solved by applying the Modular Multiplication Based Block Cipher algorithm.

The purpose of this algorithm is to increase the security of encrypted messages quickly, and cannot be reopened and if the message is reopened, it must be decrypted [33].

2. METHODS AND MATERIAL

Cryptography is part of the science or art of keeping messages safe. When a message is sent from one place to another, the contents of the message may be intercepted by other parties who are not entitled to know the contents of the message. To protect the message, the message can be converted into a code that cannot be understood by other parties [34].

In cryptography, Modular Multiplication for a Specific Modulus is quite often used in a special form, for example, in the use of prime numbers, where the modulus number is subtracted by the value 1 [35], [2]. Montgomery's modular multiplication uses the modulus N and R, R and N is the selected parameter, and usually, R is set to the power of 2 sequentially to simplify the operation of mod R [7].

Montgomery's design efficiency can be determined in terms of the area, time and energy consumed. Montgomery's modular multiplication has an iterative extension that calculates the quotient and addition of the operands followed by the shift operation. The authors [36], [37] propose 4: 2 and 5: 2 carry over to Montgomery's modular multiplication architecture by doing large word length increments and producing superior architectural savings with a throughput rate independent of word length at the cost of area requirements. which is high [38], [39], [40].

In the research, Verma et al. Resulted in a conclusion that calculating Montgomery's initial modular multiplication word approach requires basic operations to calculate the MSB (Most Significant Bit) to complete a word. The proposed method adopts the energy efficiency approach in the literature to compute the same word by all PE'S (processing elements) in all iterations. Comparing the path delay of the proposed design with other designs it has reduced the path delay, so that further research is expected to apply the proposed design to the FPGA. Also the binary exponential results of the RSA and Montgomery Powering Ladder with the proposed design will be implemented in the FPGA [41].

Efficient Interleaved Modular Multiplication (EIMM) is a parallel version of the standard alternate modular multiplication algorithm, which computes in parallel and confirms correct intermediate results using sign detection techniques. EIMM architecture is based on a sign detection technique that is responsible for determining the sign of a number represented in a number pair. EIMM calculates modular multiplication without any pre-compute phases or predefined sets of modulus [30]. EIMM has more hardware area than Modified Interleaved Modular Multiplication (MIMM), but the EIMM architecture when compared to MIMM will be more efficient in terms of performance and processing time speed, so it can be conveyed that EIMM operating time is up to 1.99x faster. from MIMM [42], [43], [44].

In a large number of modular units, the repetitive cycle operation results in a much more complex multiplication modular when compared to the addition of the subtraction modular to the structure, therefore the MAS Optimization (Multiplication Addition Subtraction) modular is focused on multiplication, the multiplication of which is divided into two types of structures namely Modular Multiplication Add-based (AMM) and Multiply-based Modular Multiplication (MMM) [45].

Cryptography uses an algorithm (cipher) and a key (key). Cipher is a mathematical function used to encrypt and decrypt data. While the key is a series of bits needed to encrypt and decrypt. Not only relying on its security on the confidentiality of the algorithm but on the confidentiality of the key [46]. Which is where in cryptography it is necessary to encrypt the text blocks $B = q1, q2,\ldots, qn$, which have n symbols of b-bit size each with a secret key [29].

The MMB method uses a 128-bit key. The key formation process in the MMB method is very simple. The key that is inputted is only divided into four (4) key sub-blocks with a length of 32 bits each. The chat message system architecture design uses built-in MMB cryptography. See Figure 1 [47].

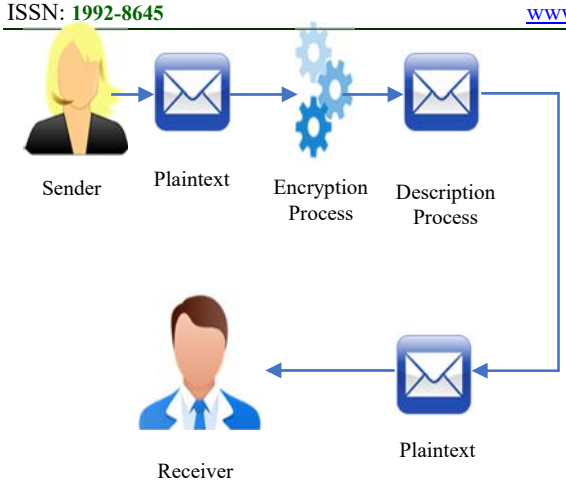

Figure 1. Design of MMB Cryptographic Chat Message Architecture

If plain text (X) or key (Y) or both are zero, then the multiplier output is zero and a nonzero input can result in output 2^{16} which is also interpreted as zero. This produces an incorrect output during the decryption process [48], then zeros must be detected and processed separately [49], If MMB has 7 rounds it is still vulnerable to different attacks [50]. The advantages of the Modular Multiplication Based Block Cipher (MMB) Method [51]:

1. In MMB, the keys used in the encryption and decryption process are the same. Meanwhile, the IDEA key used in the encryption and decryption process is not the same. The decryption key is the reverse operation of the encryption key
2. In MMB, the encryption and decryption process uses the multiplication operation modulo 2^{32} -1 so that the level of security is higher. Meanwhile, IDEA, the encryption and decryption process uses the multiplication operation modulo 2^{16} + 1.
3. In MMB, the encryption and decryption process is much faster than IDEA which only consists of 2 rounds. Meanwhile, IDEA, the encryption and decryption process is longer, consisting of 8 rounds so it is more time consuming.

Montgomery's [7] modular multiplication is an efficient modular algorithm when the modulus of n is without a specific shape. By adding a limitation to the parameter, Walter [52] proposed MMM without a conditional subtraction algorithm and usually a power of 2 for computational ease.

The XOR operation stands for Exclusive OR which consists of two (2) inputs and one (1) logical output. The XOR gate will produce a logic output of one (1) if all its inputs have different logical values. If the input logic values are the same, it will give the output logic zero (0).

The MMB method uses plaintext and a key with a length of 128 bits. In this study, we propose an algorithm that simplifies the use of MMB by directly multiplying each plain text to the key, Presented as below [53] :

- Convert plaintext to binary
- Convert key to binary
- XOR plaintext and key (Xi = Xi xor Ki)
- constant C using RoL:
- XOR the constant C with the result X, then MODULUS with 2^{32}-1 (Xi = Ci * Xi mod 2^{32}-1),
- Convert the resulting modulus to Hex,
- Convert the resulting Hex to ASCII
- Encryption= ASCII

The multiplication operation used is a multiplication operation. While the constants used can be detailed as follows:

- C = (2AAAAAAA) 16
- C0 = (025F1CDB) 16
- C1 = 2^1 * c0
- C2 = 2^2 * c0
- C3 = 2^3 * c0
- Ci =2^i * c0

4. IMPLEMENTATION OF THE ALGORITHM PROPOSED

The encryption process from the MMB method can be seen in the following example.
Suppose it is known *plaintext* = 'MAHASISWA TEKNIK' using the key generated above, the encryption process is as follows:
Plaintext : MAHASISWA TEKNIK
Key : CRYPTOGRAPHY MMB
C : 2AAAAAAA
C0 : 025F1CDB = 39787739

Convert plaintext to binary presented in table 1 below :

Table 1: Convert plaintext to binary

Plain	Decimal	Binari
M	77	01001101
A	65	01000001
H	72	01001000
A	65	01000001

S	83	01010011
I	73	01001001
S	83	01010011
W	87	01010111
A	65	01000001
	32	00100000
T	84	01010100
E	69	01000101
K	75	01001011
N	78	01001110
I	73	01001001
K	75	01001011

Convert key to binary presented in table 2 below:

Table 2: Convert key to binary

Key	Decimal	Binari
C	67	01000011
R	82	01010010
Y	89	01011001
P	80	01010000
T	84	01010100
O	79	01001111
G	71	01000111
R	82	01010010
A	65	01000001
P	80	01010000
H	72	01001000
Y	89	01011001
	32	00100000
M	77	01001101
M	77	01001101
B	66	01000010

XOR plaintext and key (Xi = Xi xor Ki) presented in table 3 below :

Table 3: XOR plaintext and key

Plain/Key/ Xi XOR Ki	Decimal	Binari
M	77	01001101
C	67	01000011
X0	14	00001110
A	65	01000001
R	82	01010010
X1	19	00010011
H	72	01001000
Y	89	01011001
X2	17	00010001
A	65	01000001
P	80	01010000
X3	17	00010001
S	83	01010011
T	84	01010100
X4	7	00000111
I	73	01001001
O	79	01001111
X5	6	00000110
S	83	01010011
G	71	01000111
X6	20	00010100
W	87	01010111
R	82	01010010
X7	5	00000101
A	65	01000001
A	65	01000001
X8	0	00000000
	32	00100000
P	80	01010000
X9	112	01110000
T	84	01010100
H	72	01001000
X10	28	00011100
E	69	01000101
Y	89	01011001
X11	28	00011100
K	75	01001011
	32	00100000
X12	107	01101011
N	78	01001110
M	77	01001101
X13	3	00000011
I	73	01001001
M	77	01001101
X14	4	00000100
K	75	01001011
B	66	01000010
X15	9	00001001

RoL is used to manipulate data in assembly language, this instruction will make the data bits shift, where RoL is the instruction to shift data to the left. In binary numbers, shifting left may mean making the binary number 2x larger.
For example: 001 when shifted to the left becomes 010, the following is presented Calculate the constant C using RoL:
C0 = 39787739
C1 = 79575478
C2 = 159150956
C3 = 318301912
C4 = 636603824
C5 = 1273207648

C6 = 2546415296
C7 = 5092830592
C8 = 10185661184
C9 = 20371322368
C10 = 40742644736
C11 = 81485289472
C12 = 162970578944
C13 = 325941157888
C14 = 651882315776
C15 = 1303764631552
Ci..n

XOR the constant C with the result X, then MODULUS with 2^{32}-1 (Xi = Ci * Xi mod 2^{32}-1), presented in table 4 below :

Table 4: XOR constant C with result X, then MODULUS with 2^{32}-1

i	Decimal Ci	Decimal Xi	Ci * Xi mod 2^32-1
0	39787739	14	557028346
1	79575478	19	1511934082
2	159150956	17	2705566252
3	318301912	17	1116165209
4	636603824	7	161259473
5	1273207648	6	3344278593
6	2546415296	20	3683665675
7	5092830592	5	3989316485
8	10185661184	0	0
9	20371322368	112	960471571
10	40742644736	28	2627719433
11	81485289472	28	960471571
12	162970578944	107	284729308
13	325941157888	3	2865897699
14	651882315776	4	484115039
15	1303764631552	9	31034028

Convert the resulting modulus to Hex, presented in table 5 below :

Table 5: Convert the resulting modulus to Hex

X	Ci * Xi mod 2^32-1	Convert to Hex
X0	557028346	213393FA
X1	1511934082	5A1E4882
X2	2705566252	A143AA2C
X3	1116165209	42875459
X4	161259473	99C9FD1
X5	3344278593	C755A441
X6	3683665675	DB90470B
X7	3989316485	EDC82385
X8	0	0
X9	960471571	393FA213
X10	2627719433	9C9FD109
X11	960471571	393FA213
X12	284729308	10F89FDC
X13	2865897699	AAD220E3
X14	484115039	1CDB025F
X15	31034028	1D98AAC

Chipertext :
213393FA5A1E4882A143AA2C4287545999C9F
D1C755A441DB90470BEDC823850393FA2139C
9FD109393FA21310F89FDCAAD220E31CDB02
5F1D98AAC

Convert the resulting Hex to ASCII, presented in table 6 below :

Table 6: Convert Hex to ASCII

X	Decimal	Hex	ASCII
X0	557028346	213393FA	!3ú
X1	1511934082	5A1E4882	ZH
X2	2705566252	A143AA2C	¡Cª,
X3	1116165209	42875459	BTY
X4	161259473	99C9FD1	Éý
X5	3344278593	C755A441	ÇU¤A
X6	3683665675	DB90470B	ÛG
X7	3989316485	EDC82385	íÈ#
X8	0	0	0
X9	960471571	393FA213	9?¢
X10	2627719433	9C9FD109	Ñ
X11	960471571	393FA213	9?¢
X12	284729308	10F89FDC	øÜ
X13	2865897699	AAD220E3	ªÒ ã
X14	484115039	1CDB025F	Û_
X15	31034028	1D98AAC	ª

So the encryption results of the proposed algorithm are:
!3úZH¡Cª,BTYÉýÇU¤AÛGíÈ#09?¢Ñ9?¢øÜªÒ ãÛª

The next is proving the description process, from the encrypted text, convert the resulting Hex to ASCII, presented in table 7 below :

Table 7: Convert ASCII to Hex and then to Decimal

X	ASCII	Hex	Decimal
X0	!3ú	213393FA	557028346
X1	ZH	5A1E4882	1511934082
X2	¡Cª,	A143AA2C	2705566252

X3	BTY	42875459	1116165209
X4	Éý	99C9FD1	161259473
X5	ÇU¤A	C755A441	3344278593
X6	ÛG	DB90470B	3683665675
X7	íÈ#	EDC82385	3989316485
X8	0	0	0
X9	9?¢	393FA213	960471571
X10	Ñ	9C9FD109	2627719433
X11	9?¢	393FA213	960471571
X12	øÜ	10F89FDC	284729308
X13	ªÒ ã	AAD220E3	2865897699
X14	Û_	1CDB025F	484115039
X15	ª	1D98AAC	31034028

Convert key to binary presented in table 8 below:

Table 8: Convert key to binary

Key	Decimal	Binari
C	67	01000011
R	82	01010010
Y	89	01011001
P	80	01010000
T	84	01010100
O	79	01001111
G	71	01000111
R	82	01010010
A	65	01000001
P	80	01010000
H	72	01001000
Y	89	01011001
	32	00100000
M	77	01001101
M	77	01001101
B	66	01000010

The next one in the final result of the description process will be generated, presented in table 9 below:

Table 9: Convert binary to plaintext

Decimal	Binari	Plain
77	01001101	M
65	01000001	A
72	01001000	H
65	01000001	A
83	01010011	S
73	01001001	I
83	01010011	S
87	01010111	W
65	01000001	A
32	00100000	
84	01010100	T
69	01000101	E
75	01001011	K
78	01001110	N
73	01001001	I
75	01001011	K

So the encryption results of the proposed algorithm are:
MAHASISWA TEKNIK

5. CONCLUSIONS

From the implementation of the algorithm proposed for data security carried out in this study, the following conclusions can be drawn:

1. The encryption process on plaintext can be done faster.
2. Keywords and key lengths are made dynamic according to the length of the plaintext, so that it makes the XOR or Modulus process easier.
3. The length of the text that has been encrypted becomes longer, so that it may be more difficult to solve.

ACKNOWLEDGMENT

We express our deepest gratitude to the Directorate of Research and Community Service (DRPM), Directorate General of Research and Development Strengthening Ministry of Research, Technology and Higher Education for funding support in the form of a Beginner Lecturer Research (PDP) grant for the 2020 fiscal year. We also express our gratitude to STMIK Kaputama and Universitas Sumatera Utara for support in the implementation of this research activity.

REFRENCES:

[1] A. M. H. Pardede, H. Manurung, and D. Filina, "Algoritma Vigenere Cipher Dan Hill Cipher Dalam Aplikasi Keamanan Data Pada File DOKUMEN," *JTIK (Jurnal Tek. Inform. Kaputama)*, vol. 1, no. 1, pp. 26–33, 2017.

[2] Z. Gu and S. Li, "A Novel Method of Modular Multiplication Based on Karatsuba-like Multiplication," 2020, doi: 10.1109/ARITH48897.2020.00014.

[3] F. Gandino, F. Lamberti, G. Paravati, J. C. Bajard, and P. Montuschi, "An algorithmic and architectural study on montgomery

exponentiation in RNS," *IEEE Trans. Comput.*, 2012, doi: 10.1109/TC.2012.84.

[4] N. Nedjah, L. M. Mourelle, M. Santana, and S. Raposo, "Massively parallel modular exponentiation method and its implementation in software and hardware for high-performance cryptographic systems," *IET Comput. Digit. Tech.*, 2012, doi: 10.1049/iet-cdt.2011.0074.

[5] G. D. Sutter, J. P. Deschamps, and J. L. Imana, "Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation," *IEEE Trans. Ind. Electron.*, 2011, doi: 10.1109/TIE.2010.2080653.

[6] X. Huang and W. Wang, "A Novel and Efficient Design for an RSA Cryptosystem with a Very Large Key Size," *IEEE Trans. Circuits Syst. II Express Briefs*, 2015, doi: 10.1109/TCSII.2015.2458033.

[7] P. L. Montgomery, "Modular Multiplication Without Trial Division," *Math. Comput.*, 1985, doi: 10.2307/2007970.

[8] A. Rezai and P. Keshavarzi, "A New CMM-NAF Modular Exponentiation Algorithm by using a New Modular Multiplication Algorithm," *Trends Appl. Sci. Res.*, 2012, doi: 10.3923/tasr.2012.240.247.

[9] S. Talapatra, H. Rahaman, and J. Mathew, "Low complexity digit serial systolic montgomery multipliers for special class of GF(2m)," *IEEE Trans. Very Large Scale Integr. Syst.*, 2010, doi: 10.1109/TVLSI.2009.2016753.

[10] Y. Y. Zhang, Z. Li, L. Yang, and S. W. Zhang, "An efficient CSA architecture for montgomery modular multiplication," *Microprocess. Microsyst.*, 2007, doi: 10.1016/j.micpro.2006.12.003.

[11] H. R. Ahmadi and A. Afzali-Kusha, "A low-power and low-energy flexible GF(p) elliptic-curve cryptography processor," *J. Zhejiang Univ. Sci. C*, 2010, doi: 10.1631/jzus.C0910660.

[12] S. R. Kuang, J. P. Wang, K. C. Chang, and H. W. Hsu, "Energy-efficient high-throughput montgomery modular multipliers for RSA cryptosystems," *IEEE Trans. Very Large Scale Integr. Syst.*, 2013, doi: 10.1109/TVLSI.2012.2227846.

[13] A. Rezai and P. Keshavarzi, "High-performance implementation approach of elliptic curve cryptosystem for wireless network applications," 2011, doi: 10.1109/CECNET.2011.5768248.

[14] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, "Systematic design of RSA processors based on high-radix montgomery multipliers," *IEEE Trans. Very Large Scale Integr. Syst.*, 2011, doi: 10.1109/TVLSI.2010.2049037.

[15] G. Sassaw, C. J. Jiménez, and M. Valencia, "High radix implementation of Montgomery multipliers with CSA," 2010, doi: 10.1109/ICM.2010.5696148.

[16] A. Rezai and P. Keshavarzi, "High-Throughput Modular Multiplication and Exponentiation Algorithms Using Multibit-Scan-Multibit-Shift Technique," *IEEE Trans. Very Large Scale Integr. Syst.*, 2015, doi: 10.1109/TVLSI.2014.2355854.

[17] G. Saldamli and Ç. K. Koç, "Spectral modular exponentiation," 2007, doi: 10.1109/ARITH.2007.34.

[18] Ç. K. Koç, T. Acar, and B. S. Kaliski, "Analyzing and comparing montgomery multiplication algorithms," *IEEE Micro*. 1996, doi: 10.1109/40.502403.

[19] P. B. McLaughlin, "New frameworks for Montgomery's modular multiplication method," *Math. Comput.*, 2003, doi: 10.1090/s0025-5718-03-01543-6.

[20] D. D. Chen, G. X. Yao, R. C. C. Cheung, D. Pao, and Ç. K. Koç, "Parameter Space for the Architecture of FFT-Based Montgomery Modular Multiplication," *IEEE Trans. Comput.*, 2016, doi: 10.1109/TC.2015.2417553.

[21] W. Dai, D. D. Chen, R. C. C. Cheung, and Ç. K. Koç, "Area-Time Efficient Architecture of FFT-Based Montgomery Multiplication," *IEEE Trans. Comput.*, 2017, doi: 10.1109/TC.2016.2601334.

[22] J. W. Bos and S. Friedberger, "Fast Arithmetic Modulo 2x py ± 1," 2017, doi: 10.1109/ARITH.2017.15.

[23] J. W. Bos and S. J. Friedberger, "Arithmetic Considerations for Isogeny-Based Cryptography," *IEEE Trans. Comput.*, 2019, doi: 10.1109/TC.2018.2851238.

[24] J. W. Bos and S. J. Friedberger, "Faster modular arithmetic for isogeny-based crypto on embedded devices," *J. Cryptogr. Eng.*, 2020, doi: 10.1007/s13389-019-00214-6.

[25] A. Karmakar, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Efficient finite field multiplication for isogeny based post quantum cryptography," 2017, doi: 10.1007/978-3-319-55227-9_14.

[26] C. Liu, J. Ni, W. Liu, Z. Liu, and M. O'Neill, "Design and Optimization of Modular

Multiplication for SIDH," 2018, doi: 10.1109/ISCAS.2018.8351082.
[27] W. Liu, J. Ni, Z. Liu, C. Liu, and M. O'Neill, "Optimized Modular Multiplication for Supersingular Isogeny Diffie-Hellman," *IEEE Trans. Comput.*, 2019, doi: 10.1109/TC.2019.2899847.
[28] F. Dewandaru and C. Rahmad, "Aplikasi Keamanan Data Menggunakan Metode Mmb Dan Lsb," *J. Inform. Polinema*, 2016.
[29] S. Krendelev, N. Zbitnev, D. Shishlyannikov, and D. Gridin, "Block cipher based on modular arithmetic and methods of information compression," 2017, doi: 10.1088/1742-6596/913/1/012009.
[30] M. A. Nassar and L. A. A. El-Sayed, "Efficient interleaved modular multiplication based on sign detection," 2016, doi: 10.1109/AICCSA.2015.7507088.
[31] A. M. Abdel Fattah, A. M. Bahaa El-Din, and H. M. A. Fahmy, "Efficient implementation of modular multiplication on FPGAs based on sign detection," 2009, doi: 10.1109/IDT.2009.5404160.
[32] M. Knežević, F. Vercauteren, and I. Verbauwhede, "Faster interleaved modular multiplication based on Barrett and Montgomery reduction methods," *IEEE Trans. Comput.*, 2010, doi: 10.1109/TC.2010.93.
[33] A. M. H. Pardede *et al.*, "Application of Message Security Application Using Vigenere Cipher Algorithm Utilizing One Time Pad (OTP) Algorithm as a Key Generator," 2019, doi: 10.1088/1742-6596/1363/1/012080.
[34] M. M. Amin, "IMAGE STEGANOGRAPHY DENGAN METODE LEAST SIGNIFICANT BIT (LSB)," *CSRID (Computer Sci. Res. Its Dev. Journal)*, 2016, doi: 10.22303/csrid.6.1.2014.53-64.
[35] P. D. Gallagher and C. Romine, "FIPS PUB 186-4 Digital Signature Standard (DSS)," *Encycl. Cryptogr. Secur.*, 2013.
[36] C. McIvor, M. McLoone, J. V. McCanny, A. Daly, and W. Marnane, "Fast montgomery modular multiplication and RSA cryptographic processor architectures," 2003, doi: 10.1109/acssc.2003.1291939.
[37] C. Mclvor, M. McLoone, and J. V. McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques," *IEE Proc. Comput. Digit. Tech.*, 2004, doi: 10.1049/ip-cdt:20040791.
[38] H. Thapliyal, A. Ramasahayam, V. R. Kotha, K. Gottimukkula, and M. B. Srinivas, "Modified montgomery modular multiplication using 4:2 Compressor and CSA adder," 2006, doi: 10.1109/DELTA.2006.70.
[39] B. Hanindhito, N. Ahmadi, H. Hogantara, A. I. Arrahmah, and T. Adiono, "FPGA implementation of modified serial montgomery modular multiplication for 2048-bit RSA cryptosystems," 2015, doi: 10.1109/ISITIA.2015.7219964.
[40] R. Verma, M. Dutta, and R. Vig, "FPGA implementation of RSA based on carry save Montgomery modular multiplication," 2016, doi: 10.1109/ICCTICT.2016.7514561.
[41] R. Verma, M. Dutta, and R. Vig, "Early-word-based montgomery modular multiplication algorithm," 2015, doi: 10.1109/SPIN.2015.7095268.
[42] J. W. Bos, P. L. Montgomery, D. Shumow, and G. M. Zaverucha, "Montgomery multiplication using vector instructions," 2014, doi: 10.1007/978-3-662-43414-7_24.
[43] P. Giorgi, L. Imbert, and T. Izard, "Parallel modular multiplication on multi-core processors," 2013, doi: 10.1109/ARITH.2013.20.
[44] P. Wang, Z. Liu, L. Wang, and N. Gao, "High radix montgomery modular multiplier on modern FPGA," 2013, doi: 10.1109/TrustCom.2013.180.
[45] J. Li, Z. Dai, W. Li, S. Yi, and S. Zhou, "Research and design of add-based length-scalable dual-field modular multiplication-addition-subtraction," 2017, doi: 10.1109/ICAM.2017.8242136.
[46] I. A. Ilyas, "Kriptografi," *Kriptografi fFle Menggunakan Metod. AES Dual Password*, 2014, doi: 10.1046/j.1364-3703.2000.00031.x.
[47] T. Wu, S. Li, and L. Liu, "Fast RSA decryption through high-radix scalable Montgomery modular multipliers," *Sci. China Inf. Sci.*, vol. 58, no. 6, pp. 1–16, Jun. 2015, doi: 10.1007/s11432-014-5215-4.
[48] S. Elagooz, N. Hamdy, K. Shehata, and M. Helmy, "Design and implementation of high and low modulo (216 + 1) multiplier used in idea algorithm on FPGA," 2003, doi: 10.1109/NRSC.2003.1217343.
[49] K. Palutla and P. Gundabathina, "Implementation of High Speed Modulo (2 +1) Multiplier for IDEA Cipher," *Procedia Comput. Sci.*, 2020, doi: 10.1016/j.procs.2020.04.216.
[50] K. Jia, J. Chen, M. Wang, and X. Wang, "Practical attack on the full MMB block cipher," 2012, doi: 10.1007/978-3-642-28496-0_11.

[51] D. Ariyus, "Pengantar Ilmu Kriptografi Teori, Analisis dan Implementasi.," *Journal of Chemical Information and Modeling*. 2008, doi: 10.1017/CBO9781107415324.004.
[52] C. D. Walter, "Montgomery exponentiation needs no final subtractions," *Electron. Lett.*, 1999, doi: 10.1049/el:19991230.
[53] R. Munir, "Pengantar Ilmu Kriptografi," *Penerbit Andi*, 2008, doi: 10.1017/CBO9781107415324.004.