Combination of RC4 and Adler32 in Improving Security and Validity

[1] **Rio Septian Hardinata,** [2] **Andysah Putera Utama Siahaan**
Faculty of Science and Technology, Universitas Pembangunan Panca Budi, Medan, Indonesia
Email: [1] rioseptian@dosen.pancabudi.ac.id, [2] andiesiahaan@gmail.com

Abstract: *Encryption is needed in the digital world. It can help users to secure confidential information that will be sent. Many algorithms can be used to do that. RC4 or Rivest Cipher 4 made by Ron Rivest is a synchronous type of stream cipher that is a cipher that has a symmetric key and encrypts or decrypts plaintext bits per bit by performing XOR operations with a number generated on S-BOX. RC4 can be run with variable key lengths and operate with byte orientation. RC4 encryption method has weaknesses. One of the disadvantages of RC4 is that the attacker can know the sample or the entire plaintext of the ciphertext without having to know the encryption key. RC4 merger with Adler32 will help improve data security. Adler32 still has weaknesses. Although these weaknesses are rare, if a third party can combine the values of A and B according to the modified results in the text, changes in information can occur even though the feature value of the hash function has the same value before and after modification. The hash function value can be solved to get the A and B values; then the information will be modified and adjusted to the A and B values. The result is that the information is no longer original while the hash function indicates that the information is still original. However, Adler32 is very helpful when combined with RC4 where both algorithms work together to improve the security and validity of a document.*

Key Words: *Adler32, RC4, encryption, decryption, combination.*

1. INTRODUCTION:

Cryptography is an absolute thing done by the sender of information so that the information is safe and not modified by specific threats. One popular algorithm in cryptography is the RC4 algorithm. RC4 is a type of cryptographic cryptography [1]–[3] This algorithm performs the encryption process by using random key rotation according to the key provided previously. Decryption will be done by entering the same key during the encryption process. However, bytes used to encrypt plaintext are not entirely random as is done by the pseudorandom generator. Ciphertext pieces can be solved by brute-force attack or can also be modified by flipping bit attacks so that the original message content changes from the original. The RC4 algorithm is one of the cryptographic algorithms that are most often used for data security [4]. This algorithm consists of 3 forming parts (S-BOX, KSA, and PRGA) to generate a keystream that will be used in the encryption and decryption process. RC4 is a type of stream cipher that has the same key in the process of encryption and decryption (symmetrical). RC4 involves the XOR process of the key generated by random. RC4 can have a key length along the number of plaintexts. However, that is impossible because it is impossible for someone to remember a very long key. Because of this, RC4 has the disadvantage of Bit-Flipping Attack or BFA where the attacker can know the sample or the entire plaintext of the ciphertext without having to know the key for the decryption process [5]. BFA can modify the contents of the ciphertext so that the decryption results will produce a different plaintext from the original plaintext. The use of the RC4 algorithm will be vulnerable if the truth on the ciphertext does not verify it before the decryption process is carried out. The use of digital signatures on RC4 algorithm sending messages is highly recommended. Many hash function methods can be performed on this algorithm. Adler32 is a useful method if applied as verification of integrity in RC4 algorithms [6]. The message that was sent before will be given a digital signature, and when the message is decrypted, the recipient of the information first checks whether the message is original or has been modified. If the digital signature produces the same value, then, of course, the message is indeed the message sent by the actual sender. Therefore, improving RC4 performance and data security from Bit Flipping Attack attacks on the encryption and decryption process is very well done [7].

2. THEORIES:

2.1 RC4

RC4 is an encryption algorithm created by Ronald Rivest from RSA Security. It is used in WEP and WPA, whose encryption protocol is commonly used on wireless routers [8]. How RC4 works basically, but the code was leaked to the internet in 1994. RC4 was initially very widely used because of its simplicity and speed. Usually, 16 bytes are used for strong encryption, but shorter key lengths are also widely used because of restrictions. Over time this code is shown to produce an output bias towards a particular sequence, especially in the first few bytes of the generated keystream [9]. Lots of development come from the RC4 algorithm [10]. This development caused a future version of the RC4 code that is more widely used today, called RC4-drop [n], where the first n byte of the keystream is dropped to get rid of this bias output. Some important uses of RC4 are implemented in some software such as Microsoft Excel, Adobe Acrobat 2.0 (1994), and BitTorrent clients. It is necessary to declare a key to start the RC4 encryption process. This key has a length between 40-bit and 256-bit. A 40-bit key is a five-character ASCII code that will be translated into 40 binary characters [11].

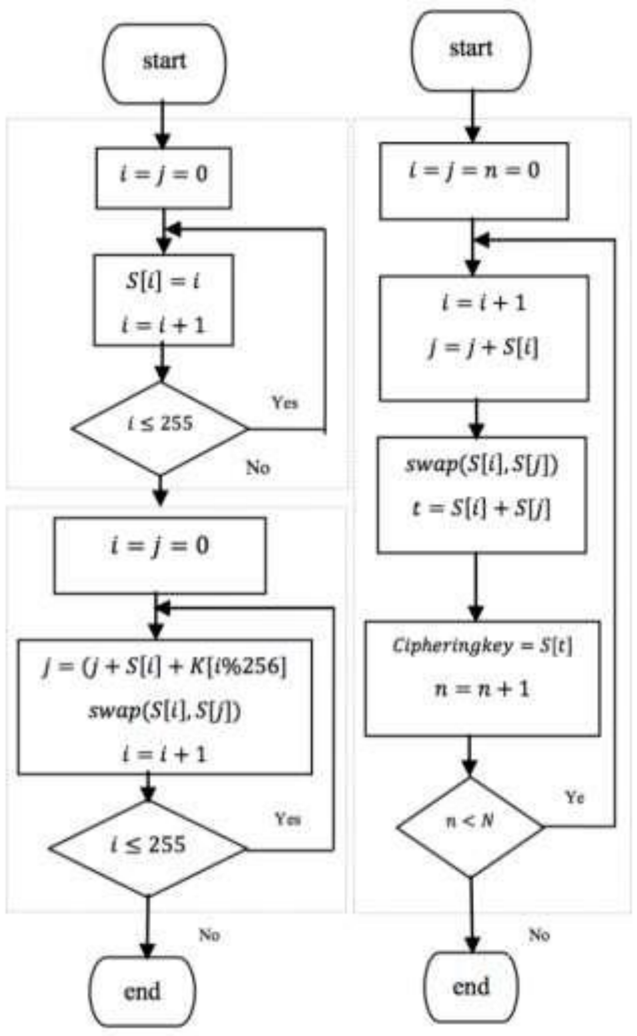

Figure 1. RC4 flowchart [12]

The RC4 algorithm has three stages: S-Box initialization, initial permutation of the S-box element and finally key flow generation (PRGA). All three stages must be carried out for each new key. The RC4 algorithm shown in Figure 1 uses a variable length key of 1 byte to 256 bytes, and the key used to initialize a 256-byte array. Arrays are used to obtain random values from the next generation and then generate a stream of pseudorandom keys, which are then XORed against plaintext or ciphertext [13].

2.2 Adler32

Adler32 is a checksum designed to avoid corruption in Zlib compressed data or corruption in algorithms. Because Adler32 is much faster than a more reliable competitor, CRC32, this method is better [14]. If compressed data does not match the Adler32 checksum, the application can inform the protocol or user to retransmit data [15]. However, Adler32's primary use is the Zlib debug implementation. Adler32 and CRC32 are insufficient for any purpose that requires a high degree of accuracy. Irresponsible parties can try to steal data that is being transmitted. Adler32 can verify data to ensure the integrity of the information. Adler-32 checksum is obtained by calculating two 16-bit checksums A and B and combining the results of 16 bits into a 32-bit integer [16]. A is the sum of all bytes plus one, and B is the sum of individual A values from each step. Initially, A is initialized to 1, B to 0. The number is obtained by modulo 65521, the most significant prime number smaller than 216. Bytes are stored in sequence and B occupies the two most significant bytes [17]. This function can be expressed as:

$$A = 1 + D_1 + D_2 + ... + D_n \pmod{65521}$$
$$B = (1 + D_1) + (1 + D_1 + D_2) + ... + (1 + D_1 + D_2 + ... + D_n) \pmod{65521}$$

$= n\times D_1 + (n-1)\times D_2 + (n-2)\times D_3 + \ldots + D_n + n \pmod{65521}$

$Adler\text{-}32(D) = B \times 65536 + A$

3. RESULT AND DISCUSSION:

RC4 security is guaranteed and recognized, but the strength of this algorithm is in the complexity of the keys used. If the key used is straightforward, then the possibility of plaintext will be solved. Problems that occur are not key. Sometimes a third party does not need a key to destroy information. To do RC4 modification does not require a key, just by modifying the bit. By modifying the bit, the integrity of this information has changed. The following is the mechanism of the method that will be performed on the plaintext that will be encrypted.

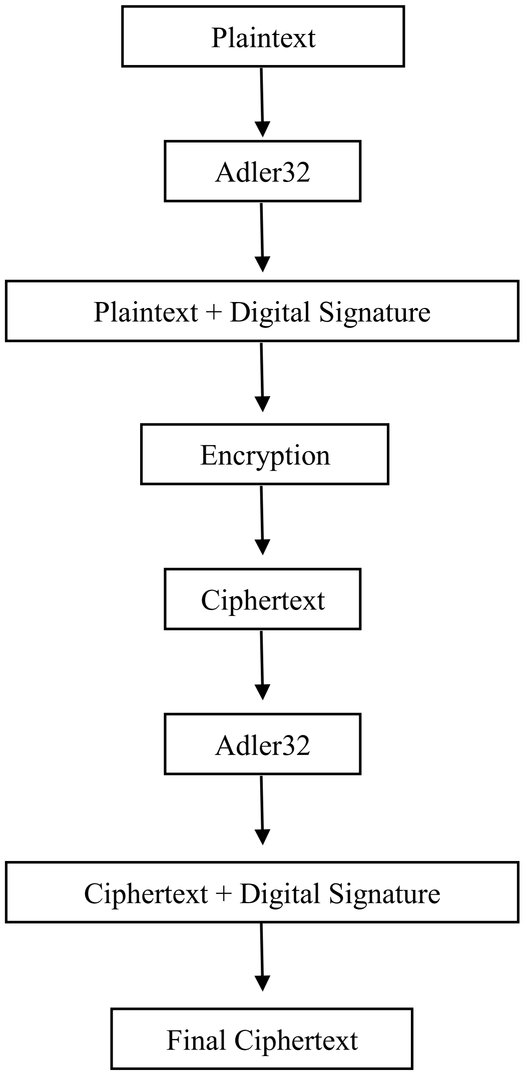

Figure 2. RC4 and Adler32 Encryption Mechanism

Figure 2 describes the encryption process from RC4. In the encryption process, the first plaintext will be calculated the value of its integrity. This value is calculated using the Adler32 algorithm. The value of Adler32 will be combined with plaintext. So there is a digital signature that signifies the plaintext. Between plaintext and digital signature will be given a delimiter or tag so that it can be easily separated during the decryption process. The ciphertext results are then re-done with the digital signature process so that the ciphertext is the result of encryption from the plaintext. After the initial ciphertext is obtained, the ciphertext will be combined with a digital signature, and then the re-encryption process will be carried out. There are two encryption processes done on the plaintext. After the ciphertext and digital signature are encrypted, the last encryption result is the value of the last ciphertext. During the encryption process of this method, there is one encryption process, and two also process the digital signature calculation. All of these processes are carried out in order to maintain the integrity of the plaintext that will be sent. The use of keys is done only once during the encryption process. The key used in each encryption may use the same or different key.

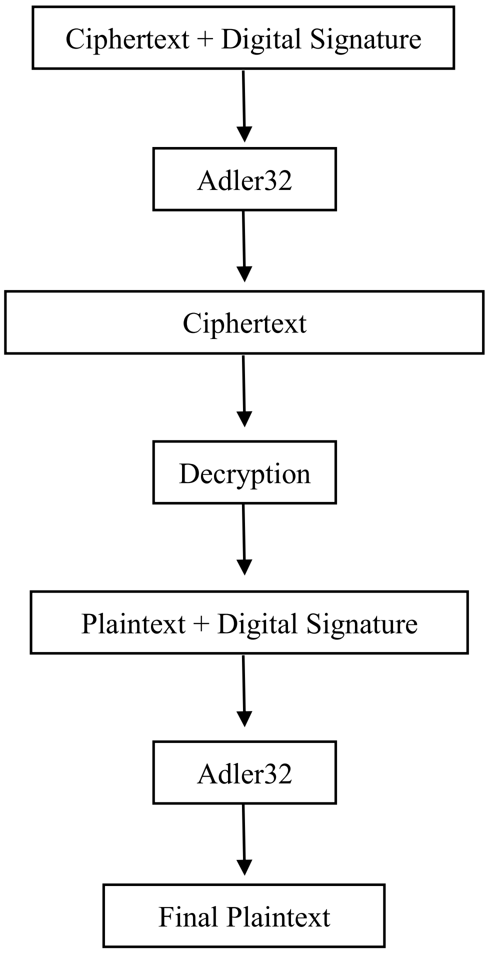

Figure 3. RC4 and Adler32 Decryption Mechanism

Figure 3 describes the decryption process of RC4. In this decryption process, the process carried out is the opposite of the encryption process. The first ciphertext will be separated from the digital signature, and the value of the digital signature will be calculated whether it matches the ciphertext. Moreover, type the value by the ciphertext, the ciphertext will be decrypted. The decryption result is plaintext and the digital signature of the plaintext. The resulting plaintext is a merge of plaintext and digital signature. The value taken from the digital signature will then be calculated whether it is the same as the previous plaintext. During the decryption process of this method, there is one decryption process, and two also process the digital signature calculation as done in the encryption process. This process is also done so that the ciphertext can return to plaintext as before. The key that is also used once when the decryption process is first performed.

4. RESULT AND DISCUSSION

4.1 Adler32 Initialization

In this section, an example of testing the plaintext will be explained. The experimental plaintext is "FAKULTAS ILMU KOMPUTER." It consisting of 22 characters; it will be given a hash function to ensure no changes to the information that was first created. Here are the explanations and calculations to get a value from Adler32.

Table 1. Adler32 calculation

Char	Byte	A	B
F	70	A = 1 + 70 = 71	B = 0 + 71 = 71
A	65	A = 71 + 65 = 136	B = 71 + 136 = 207
K	75	A = 136 + 75 = 211	B = 207 + 211 = 418
U	85	A = 211 + 85 = 296	B = 418 + 296 = 714
L	76	A = 296 + 76 = 372	B = 714 + 372 = 1086
T	84	A = 372 + 84 = 456	B = 1086 + 456 = 1542
A	65	A = 456 + 65 = 521	B = 1542 + 521 = 2063
S	83	A = 521 + 83 = 604	B = 2063 + 604 = 2667

	32	A = 604 + 32 = 636	B = 2667 + 636 = 3303
I	73	A = 636 + 73 = 709	B = 3303 + 709 = 4012
L	76	A = 709 + 76 = 785	B = 4012 + 785 = 4797
M	77	A = 785 + 77 = 862	B = 4797 + 862 = 5659
U	85	A = 862 + 85 = 947	B = 5659 + 947 = 6606
	32	A = 947 + 32 = 979	B = 6606 + 979 = 7585
K	75	A = 979 + 75 = 1054	B = 7585 + 1054 = 8639
O	79	A = 1054 + 79 = 1133	B = 8639 + 1133 = 9772
M	77	A = 1133 + 77 = 1210	B = 9772 + 1210 = 10982
P	80	A = 1210 + 80 = 1290	B = 10982 + 1290 = 12272
U	85	A = 1290 + 85 = 1375	B = 12272 + 1375 = 13647
T	84	A = 1375 + 84 = 1459	B = 13647 + 1459 = 15106
E	69	A = 1459 + 69 = 1528	B = 15106 + 1528 = 16634
R	82	A = 1528 + 82 = 1610	B = 16634 + 1610 = 18244

In Adler32 search calculations, first, the value of A is initialized with 1 and B with 0. Each character will be converted into bytes to calculate accumulation in the search for values A and B. First, the character tested is an F character with a byte value of 70. A is obtained by adding the previous value with 70, which is 1 + 70 so that it produces a value of 71. It is done until the entire 22 characters have been calculated. The final value of A is 1610 and B is 18244. This value cannot be used yet and must be changed to hexadecimal values first. This value must be done modulo to 65521, the largest prime number at 16 bits.

$$\begin{aligned} A &= A \text{ Mod } 65521 \\ &= 1610 \text{ Mod } 65521 \\ &= 1610 \end{aligned}$$

$$\begin{aligned} B &= B \text{ Mod } 65521 \\ &= 18244 \text{ Mod } 65521 \\ &= 18244 \end{aligned}$$

$$\begin{aligned} N &= B * 65536 + A \\ &= 18244 * 65536 + 1610 \\ &= 1195640394 \end{aligned}$$

This value is still in the form of a 32-bit integer value, so it needs to be converted to hexadecimal numbers. The result is 4744064A. This is the value of the hash function for the plaintext. This value will be combined with the plaintext which will then be encrypted using the RC4 algorithm. So the plaintext changes to "FAKULTAS ILMU KOMPUTER4744064A". Addition of 8 characters will occur at the end of the plaintext.

4.2 Encryption Process

S-Box formation is done to get the key pair that will be given the XOR process to the plaintext. The key is obtained by searching based on the pseudorandom number based on the S-Box that has been formed previously. The following table is the result of the encryption process against the plaintext that has been added to the hash function.

Table 2. RC4 encryption process

PT	ASCII	Key	ASCII	CT
F	70	212	146	’
A	65	171	234	ê
K	75	237	166	¦
U	85	103	50	2
L	76	151	219	Û
T	84	14	90	Z
A	65	161	224	à

S	83	134	213	Õ
	32	104	72	H
I	73	154	211	Ó
L	76	136	196	Ä
M	77	244	185	¹
U	85	255	170	ª
	32	46	14	
K	75	172	231	ç
O	79	158	209	Ñ
M	77	36	105	i
P	80	77	29	
U	85	204	153	™
T	84	212	128	€
E	69	139	206	Î
R	82	37	119	w
4	52	25	45	-
7	55	133	178	²
4	52	6	50	2
4	52	209	229	å
0	48	235	219	Û
6	54	113	71	G
4	52	221	233	é
A	65	172	237	í

Table 2 is the result of the encryption process using the RC4 algorithm. At first, plaintext totaling 22 characters will increase to 30 characters after being inserted by the hash function at the end of the plaintext. Each character on the plaintext will be converted to a byte value which will then be XORed with a list of keys to get ASCII from the ciphertext. In the end, each ASCII will be returned to the character form to get the ciphertext.

4.3 Addition of Adler32 After Encryption

In this section, the addition of the hash function will be carried out at the end of the ciphertext previously obtained. The ciphertext that is obtained is "ê¦2ÛZàÕHÓÄ¹ªçÑi ™ ?? Îw-²2åÛGéí". This ciphertext consists of 30 characters will be given a hash function to ensure that there is no change in information after the encryption process is performed. The following is an explanation and calculation to get a value from Adler32 after the encryption process.

Table 3. Adler32 calculation after encryption

Char	A	B
¦ 166	A = 381 + 166 = 547	B = 528 + 547 = 1075
2 50	A = 547 + 50 = 597	B = 1075 + 597 = 1672
Û 219	A = 597 + 219 = 816	B = 1672 + 816 = 2488
Z 90	A = 816 + 90 = 906	B = 2488 + 906 = 3394
à 224	A = 906 + 224 = 1130	B = 3394 + 1130 = 4524
Õ 213	A = 1130 + 213 = 1343	B = 4524 + 1343 = 5867
H 72	A = 1343 + 72 = 1415	B = 5867 + 1415 = 7282
Ó 211	A = 1415 + 211 = 1626	B = 7282 + 1626 = 8908
Ä 196	A = 1626 + 196 = 1822	B = 8908 + 1822 = 10730
¹ 185	A = 1822 + 185 = 2007	B = 10730 + 2007 = 12737
ª 170	A = 2007 + 170 = 2177	B = 12737 + 2177 = 14914
14	A = 2177 + 14 = 2191	B = 14914 + 2191 = 17105

ç	231	A = 2191 + 231 = 2422	B = 17105 + 2422 = 19527
Ñ	209	A = 2422 + 209 = 2631	B = 19527 + 2631 = 22158
i	105	A = 2631 + 105 = 2736	B = 22158 + 2736 = 24894
	29	A = 2736 + 29 = 2765	B = 24894 + 2765 = 27659
™	153	A = 2765 + 153 = 2918	B = 27659 + 2918 = 30577
€	128	A = 2918 + 128 = 3046	B = 30577 + 3046 = 33623
Î	206	A = 3046 + 206 = 3252	B = 33623 + 3252 = 36875
w	119	A = 3252 + 119 = 3371	B = 36875 + 3371 = 40246
-	45	A = 3371 + 45 = 3416	B = 40246 + 3416 = 43662
²	178	A = 3416 + 178 = 3594	B = 43662 + 3594 = 47256
2	50	A = 3594 + 50 = 3644	B = 47256 + 3644 = 50900
å	229	A = 3644 + 229 = 3873	B = 50900 + 3873 = 54773
Û	219	A = 3873 + 219 = 4092	B = 54773 + 4092 = 58865
G	71	A = 4092 + 71 = 4163	B = 58865 + 4163 = 63028
é	233	A = 4163 + 233 = 4396	B = 63028 + 4396 = 67424
í	237	A = 4396 + 237 = 4633	B = 67424 + 4633 = 72057

In Adler32 search calculations, first, the value of A is initialized with 1 and B with 0. Each character will be converted into bytes to calculate accumulation in the search for values A and B. First, the character tested is an F character with a byte value is 166. A is obtained by adding the previous value with 166, which is 1 + 166 so that it produces a value of 167. It is done until the entire 30 characters have been calculated. The final value of A is 4633 and B is 72057. This value cannot be used yet and must be converted to a hexadecimal value first. This value must be done modulo to 65521, the largest prime number at 16 bits.

$$\begin{aligned} A &= A \text{ Mod } 65521 \\ &= 4633 \text{ Mod } 65521 \\ &= 4633 \end{aligned}$$

$$\begin{aligned} B &= B \text{ Mod } 65521 \\ &= 72057 \text{ Mod } 65521 \\ &= 4536 \end{aligned}$$

$$\begin{aligned} N &= B * 65536 + A \\ &= 4536 * 65536 + 4633 \\ &= 428347929 \end{aligned}$$

This value is still in the form of a 32bit integer value, so it needs to be converted to hexadecimal numbers. The result is 19881219. It is the value of the hash function for the ciphertext that will be combined with the ciphertext. The ciphertext will change to "' ê¦2ÛZàÕHÓÄ¹ªçÑi ™ ?? Îw-²2åÛGéí19881219". Addition of 8 characters will occur at the end of the ciphertext so that the total characters are 38 characters. The use of hash functions in the encryption process is very well done so that information that has been successfully resolved by a third party can be checked for correctness. The information can be modified if it is successfully dismantled and will be sent back to the destination. If this happens, the value of the hash function sign will change, and it can be ascertained that the information is no longer authentic.

5. CONCLUSION:

The use of the RC4 algorithm as an encryption process has good speed. However, this algorithm has weaknesses too. RC4 has the disadvantage of Bit-Flipping Attack. By reversing the bit value, it gives the result of plaintext without having to know the key; this will produce the plaintext of the ciphertext. Modification of information can occur if the plaintext is successfully resolved. The use of hash functions is excellent to ensure that the information is based on the right party. Adler32 can be used to provide the correct information to be transmitted. This algorithm also has good speed in calculating the values of A and B to be used as hexadecimal values as a feature value of the hash function.

REFERENCES:

1. A. P. U. Siahaan, "RC4 Technique in Visual Cryptography RGB Image Encryption."

2. H. Ming and S. LiZhong, “A New System Design of Network Invasion Forensics,” in *2009 Second International Conference on Computer and Electrical Engineering*, 2009, pp. 596–599.
3. A. Lubis and A. P. U. Siahaan, “Network Forensic Application in General Cases,” *IOSR J. Comput. Eng.*, vol. 18, no. 6, pp. 41–44, 2016.
4. I. Sumartono, A. P. U. Siahaan, and N. Mayasari, “An Overview of the RC4 Algorithm,” *IOSR J. Comput. Eng.*, vol. 18, no. 6, pp. 67–73, 2016.
5. A. P. U. Siahaan, “Blum Blum Shub in Generating Key in RC4,” *Int. J. Sci. Technoledge*, vol. 4, no. 10, pp. 1–5, 2016.
6. A. P. U. Siahaan, “Adler-32 Integrity Validation in 24bit Color Image.”
7. D. Kurnia, H. Dafitri, and A. P. U. Siahaan, “RSA 32-bit Implementation Technique,” *Int. J. Recent Trends Eng. Res.*, vol. 3, no. 7, pp. 279–284, 2017.
8. W. Stallings, *Cryptography and Network Security Principles and Practices*, 4th ed. Prentice Hall, 2005.
9. A. P. U. Siahaan, *How to Code: Advanced Encryption Standard in C#*. Medan: Fakultas Ekonomi Universitas Panca Budi, 2018.
10. A. Putera Utama Siahaan, E. Elviwani, and B. Oktaviana, “Comparative Analysis of RSA and ElGamal Cryptographic Public-key Algorithms,” in *Proceedings of the Joint Workshop KO2PI and The 1st International Conference on Advance & Scientific Innovation*, 2018.
11. T. D. B. Weerasinghe, “An effective RC4 stream cipher,” in *2013 IEEE 8th International Conference on Industrial and Information Systems*, 2013, pp. 69–74.
12. A. A. Okedola and Y. N. Asafe, “No TitleRSA and RC4 Cryptosystem Performance Evaluation Using Image and Text File,” *Int. J. Sci. Eng. Res.*, vol. 6, no. 5, p. 289, 2015.
13. Y. Kumar, R. Munjal, and H. Sharma, “Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures,” *Int. J. Comput. Sci. Manag. Stud.*, vol. 11, no. 3, pp. 60–63, 2011.
14. G. Castagnoli, S. Brauer, and M. Herrmann, “Optimization of cyclic redundancy-check codes with 24 and 32 parity bits,” *IEEE Trans. Commun.*, vol. 41, no. 6, pp. 883–892, Jun. 1993.
15. R. Damasevicius, G. Ziberkas, V. Stuikys, and J. Toldinas, “Energy Consumption of Hash Functions,” *Electron. Electr. Eng.*, vol. 18, no. 10, Dec. 2012.
16. P. Koopman, “32-bit cyclic redundancy codes for Internet applications,” in *Proceedings International Conference on Dependable Systems and Networks*, pp. 459–468.
17. S. Sanguanpong, “COMPARISON OF HASH STRATEGIES FOR FLOW-BASED LOAD BALANCING,” *Int. J. Electron. Commer. Stud.*, vol. 6, no. 2, pp. 259–268, Dec. 2015.