Multidimensional key RC6 algorithm

Conference Paper · January 2019

DOI: 10.1145/3309074.3309095

3 authors:

Rommel Evan J Paje
Bicol University

Ariel M. Sison
Emilio Aguinaldo College

Ruji Medina
Technological Institute of the Philippines

Multidimensional Key RC6 Algorithm

Rommel Evan J. Paje
Bicol University
Legaspi City
rejpaje@bicol-u.edu.ph

Ariel M. Sison
Technological Institute of the Philippines- Graduate Programs
Quezon City, Manila
ariel.sison@eac.edu.ph

Ruji P. Medina
Technological Institute of the Philippines- Graduate Programs
Quezon City, Manila
ruji.medina@tip.edu.ph

ABSTRACT
Data confidentiality grows equally important along with technological advancement. Encryption technique is an essential aspect of information security. The security of encryption technique relies on its key size used. Hence, a longer key in an encryption algorithm will be harder to break compared to an algorithm using a smaller key. This paper modified the RC6 Algorithm using a multidimensional key size to increase its security. A key size was selected using 1024, 1280, 1792, 2048, and 2816bits to vary the level of security of data that is being encrypted which caters an additional layer of security to brute force attack such as exhaustive search.

CCS Concepts
• **Security and privacy→Key management**

Keywords
Encryption; Exhaustive Search; Key Length; Multidimensional; Security;

1. INTRODUCTION
With the advancement of both computer and internet technology, multimedia data such as images, audio, video are being used more extensively. Information security and privacy issues are becoming more important with the continuous growth of technology.

Encryption technique is one of the most essential aspects that prove useful to secure confidential information. Security of an encryption usually consists of its perceptual security, its key space, key sensitivity, and its ability against potential attacks. (1) Perceptual security is when a method is used to encrypt a datum; for example, if an encrypted image is not perceptually recognized, the encryption is secure in perception. (2) Keyspace is generally defined as the number of encryption keys that are available in the cryptosystem. (3) Key sensitivity is an ideal encryption and is sensitive with respect to the secret key i.e. the change of a single bit in the secret key should produce a completely different encrypted result. Therefore, security of encryption algorithm should be secure in perception, have large key space, high key sensitivity, and resist potential attacks[1].

A key is a numeric or alphanumeric text or a special symbol. The Key is used at the time the encryption takes place on the plaintext and at the time the decryption takes place on the ciphertext. Selecting a key is of vital importance since encryption algorithm security depends directly on it. The strength of the encryption algorithm relies on the secrecy of the key, length of the key, the initialization vector, and how they all work together[2].

Designing a strong encryption algorithm is crucial considering the present pace of technological advancement. The power of the computer is growing every day; the attackers of the currently implemented known security are continuously evolving and varying their attacks. Outdated security would be vulnerable. It is important to understand and take advantage of the algorithms that are resistant against various cryptanalytic and brute force attack in order to provide better security to various real-life computing applications.

RC6 Algorithm had considered dozens of alternatives and subjected to intense cryptanalysis to achieve three goals: high security, exceptional simplicity, and good performance during its development[3]. It provides a great amount of flexibility with regards to the key, the number of rounds, and the word size of the algorithm. Its simplicity allowed analysts to quickly refine and improve the estimates of its security during its development[4]. With its simplicity in structure, there are variants of modification of the algorithm which was developed by increasing its key size[4][5][6].

Other algorithms also ventured by enhancing its key size as their modification such as; modified DES and AES using 1024 bit key[7][8], an expanded 128-bit DES Algorithm[9], and the 512 Bit Key AES Algorithm [10].

Encryption algorithms integrity heavily depends on the size of the key. An encryption algorithm using longer key is harder to break than the one using a smaller key. If a weak key is used in an algorithm, an intruder could easily decrypt the data. In such, RC6 Algorithm is vulnerable to attacks such as differential and brute force attack or exhaustive search if the key size is small [11][12]. A desirable property of an encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext[13].

This research proposes a modification of the RC6 Algorithm by using a multidimensional key for encryption and decryption for enhanced security. The algorithm is based on five levels of security using different sizes of the input key. The modification also includes adding additional registers and additional transformation function on the encryption and decryption process and increasing the word size from 32 bit to 64 bit word size.

ICCSP 2019, January 19–21, 2019, Kuala Lumpur, Malaysia

ACM ISBN 978-1-4503-6618-2/19/01…$15.00

DOI: https://doi.org/10.1145/3309074.3309095

2. THE RC6 ALGORITHM

RC6 is more accurately specified as RC6-w/r/b where w is the word size in bits, r is the number of rounds, and b denotes the length of the encryption key in bytes.RC6 has a proper block size of 128 bits and supports key sizes of 128, 192, and 256 bits[14]. The RC6 algorithm is a Feistel algorithm whose data are mixed via data-depended rotations. The RC6 has four registers with a 32-bit length that help in performing rotation [15]. Figure 1 shows the RC6 algorithm consists of three components: a key expansion algorithm, an encryption an algorithm, and a decryption algorithm[16].

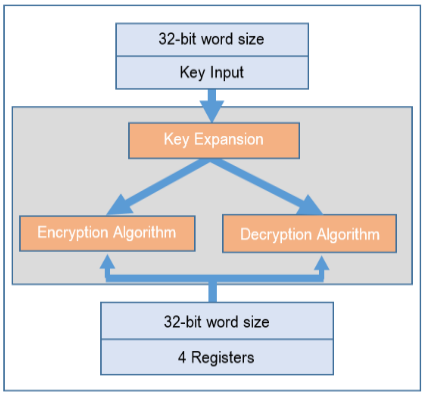

Figure 1. RC6 Algorithm Structure

2.1 The Key Expansion

```
Key Expansion – Standard RC6 Algorithm
Input
        b = User supplied key loaded in array L[ ]
        r = Number of rounds
        Pw = Odd((e-2)2^w) = B7E15163
        Qw = Odd((θ-2)2^w) = 9E379B9
Output
        w-bit round keys S[0,.., 2r+3]
Algorithm
        S[0] = Pw
        for i = 1 to 2r+3 do
                S[i] = S[i-1] + Qw
        X=Y=i=j0
        v = 3 * max(c, 2r+4)
        for j=1 to Iteration do
        {
                X = S[i] = (S[i] + X + Y) <<< 3
                Y = L[j] = (L[j] + X + Y) <<< (X+Y)
                i = (i+1) mod (2r + 4)
                j = (j+1) mod c
        }
```

Figure 2. Key Expansion Algorithm

The key expansion algorithm is used to expand the user-supplied key to fill an array S. The user must supply a key of b bytes, and from which (2r+4) words are derived and stored in a round key array S. The key bytes are loaded into an array L[0,..,c-1] of c=ceil(b/u) where u=w/8 in little-endian order. Any unfilled byte positions in L are zeroed[17]. The (2r+4) derived words are stored in array S[0,..,2r+3] for later encryption or decryption process[18]. The key schedule also uses a magic constant of P_w and Q_w[17]as shown in Table 1.

Table 1. Magic Constant Values P_w and Q_w

W	16	32	64
P_w	B7E1	B7E15163	B7E151628AED2A6B
Q_w	9E37	9E3779B9	9E3779B97F4A7C15

2.2 Encryption Algorithm

The process of encryption and decryption are both composed of three stages: pre-whitening, an inner loop of rounds, and post-whitening. The block encryption process works with four w-bit registers A, B, C, D which contain the plaintext as well as the ciphertext at the end of the encryption. The first registers B and D undergo pre-whitening. Next, there are r-rounds. The registers B and D are put through the quadratic equation which is the transformation function of the process. Finally, registers A and C undergo a post-whitening process.

```
Encryption Algorithm – Standard RC6 Algorithm
Input
        Plaintext stored in four w-bit input registers A, B, C, D
        Number r of rounds
        w-bit round keys S[0,..,2r+3]
Output
        Cipher text stored in A, B, C, D
Algorithm
        B = B + S[0]
        D = D + S[1]
        for I = 1 to r do
        {
                t = (B * (2B + 1)) <<< log w
                u = (D * (2D + 1)) <<< log w
                A = ((A ⊕ t) <<< u) + S[2i]
                C = ((C ⊕ u) <<< t) + S[2i + 1]
                (A, B, C, D) = (B, C, D, A)
        }
        A = A + S[2r + 2]
        C = C + S[2r + 3]
```

Figure 3. Encryption Algorithm

The decryption process is similar to the overall structure of the encryption process. However, the procedure begins with pre-whitening steps for C and A. The loop runs in reverse for the number of r rounds. After completing the loop, registers D and B will undergo a post-whitening step.

3. PROPOSED MODIFICATION OF THE RC6 ALGORITHM

The modification was done in order for RC6 Algorithm to accommodate a higher key. In this modification, the RC6 algorithm structure in the key expansion was not redesigned but rather, increased in its key-size input and the word size to provide a higher level of security.

The word size was modified from the original 32 bit to become 64 bit. It also included increasing the number of registers from the original of 4 registers to 16 registers to accommodate higher input block during the encryption and decryption processes. Each register has a 64-bit word size.

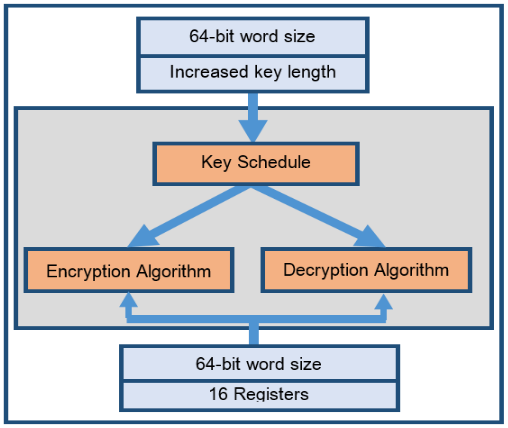

Figure 4. Modified RC6 Algorithm Structure

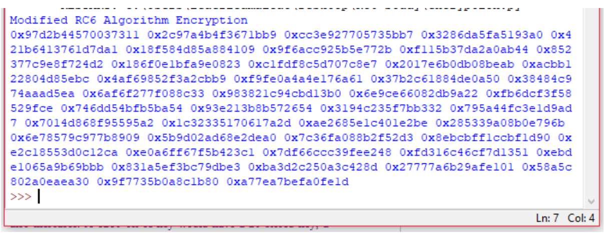

Figure 5. Key Expansion of RC6 Algorithm

4. SIMULATION OF THE MODIFIED RC6 ALGORITHM

The modified algorithm was developed using Python in an Intel® Core ™ i3-3220 CPU @ 3.3GHz and 4GB RAM in Windows 7 computer.

The modified algorithm was tested on a number of bit key that would be accommodated by the modified algorithm. According to the importance of the message, a suitable length of a key will be selected; there are five levels of selection that would vary according to the degree of security of the message that will be decided by the user.

1. 1024 bit key
2. 1280 bit key
3. 1792 bit key
4. 2048 bit key
5. 2816 bit key

4.1 Key Expansion

A 2816 bit key input was used together with a word size of 64bit for simulation.

A. User Key

2816bit key was stored in L[ ] array and will be used on the key generation.

B. Generated Key

The generated keys stored on array S[] will be used during the encryption process of the RC6 Algorithm.

Key Expansion – Modified RC6 Algorithm

Input

b byte key that is preloaded into c word array L[0,1..c-1]
r denotes the number of rounds
P_w = Odd((e-2)2^w) = B7E151628AED2A6B
Q_w = Odd((θ-2)2^w) = 9E379B97F4A7C15

Output

(2r+4) w-bit round keys S[0,1,..,2r+2, 2r+3]

Algorithm

1. S[0] = P_w
2. Repeat step 3 for i = 1 to 2r+3 do
3. S[i] = S[i-1] + Q_w
4. X=Y=a=b=0
5. Iteration = 3 * max(c,2r+4)
6. Repeat Step 7 to 10 for j=1 to Iteration do
7. X = S[a] = (S[a] + X + Y) <<< 3
8. Y = L[b] = (L[b] + X + Y) <<< (X+Y)
9. a = (a+1) mod (2r + 4)
10. b = (b+1) mod c

Figure 6. Key Expansion Algorithm

Figure 7. User Input Key

5. RESULTS AND DISCUSSIONS

5.1 Analysis based on Execution Time

Table 2 shows the execution time of the Standard RC6 and the modified RC6 with five key size used. The data used for this analysis were audio, document, image, text file, video, excel, pdf, and a power point files.

Table 2. Execution Time

File Type	Size (bytes)	Key Sizes (bit)					
		128	1024	1280	1792	2048	2816
amr	731878	23.8682	9.2041	9.2977	9.2977	9.3913	9.2197
docx	195770	6.5988	2.7144	2.6988	2.7144	2.7768	2.7768
jpg	2021952	64.7716	24.5858	24.3050	24.1958	24.0710	24.1490
txt	12267	0.7800	0.5460	0.5148	0.4680	0.4992	0.4212
mp4	3669870	114.9727	43.5087	43.7115	43.6959	43.3059	43.5711
xlsx	84919	2.9952	1.3884	1.3416	1.3416	1.3884	1.3572
pdf	360958	12.1057	4.7112	4.6488	5.0076	4.8048	5.0232
pptx	7384560	237.9951	89.4510	90.4182	90.1530	88.5462	88.7646
Total	14462174	464.0874	176.1095	176.9363	176.8739	174.7835	175.2827

This shows that the RC6 with five keys used in this study is superior in terms of its execution time. This is because the word sizes of each registers were increased from 32 bit to 64 bit and used 16 registers from that of the standard with 4 registers.

5.2 Analysis based on Iteration

Table 3 shows the number of iteration of each algorithm to process the given file size. The modified RC6 algorithm accommodates 128 bytes on every iteration while the standard RC6 has only 16 bytes in every iteration. Due to the use of 16 registers of the modified RC6, and the 64 bit word size of each register, it is clearly noticeable that the modified RC6 has a lesser number of iteration to finish the given file.

Table 3. Total Iteration

File Type	Size (bytes)	**Key Sizes (bit)**					
		128	1024	1280	1792	2048	2816
		Registers Size (byte)					
		16	128	128	128	128	128
amr	731878	45743	5718	5718	5718	5718	5718
docx	195770	12236	1530	1530	1530	1530	1530
jpg	2021952	126372	15797	15797	15797	15797	15797
txt	12267	767	96	96	96	96	96
mp4	3669870	229367	28671	28671	28671	28671	28671
xlsx	84919	5308	664	664	664	664	664
pdf	360958	22560	2820	2820	2820	2820	2820
pptx	7384560	461535	57692	57692	57692	57692	57692
Total	14462174	903888	112988	112988	112988	112988	112988

5.3 Analysis based on Throughput

The throughput was computed by dividing the total execution time to the total number of file size that was used.

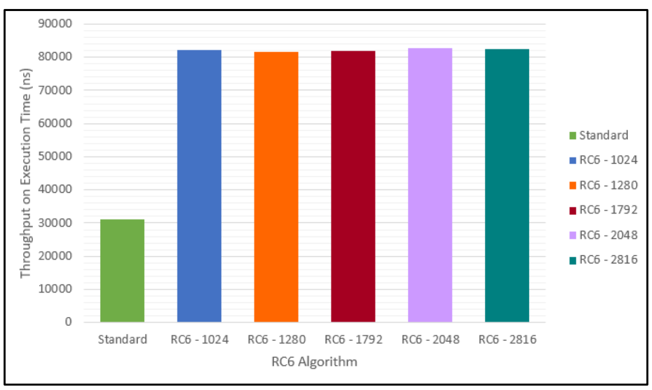

Figure 8. Throughput on Execution Time

Figure 8 shows the throughput on the Standard RC6 and the modified RC6. This shows that the modified RC6 has a better throughput compared to the standard RC6. This is due to the use of 64bit word size per register and using 16 registers. This means that, the modified RC6 has 1024 bit per iteration compared to the standard with only 128 bit per iteration.

5.4 Performance Analysis

The performance analysis of the Standard RC6 and the modified RC6 algorithm together with the key size used was measured by using the memory used by algorithm over the amount of data that was processed in a given time.

Given the computer specification and the programming language used, the Standard RC6 Algorithm used 6kb of memory space and the Modified RC6 Algorithm used 8kb of memory space which both includes the instruction code and user key.

The computation done was based from [19] by running the code to count the number of iteration in a particular time. In this case, Table 4 shows the test that was done. The data were obtained using the audio (.amr) file as an input data which has a size of 731,878bytes. The average number of iteration at 1.01400ns in 10 times of execution were shown. Regardless of the key size used, the number of iteration of the modified RC6 algorithm was closer to each other compared to the standard RC6 Algorithm.

Table 4. Iteration in Given Time

Time (ns)	**Standard**	**RC6-1024**	**RC6-1280**	**RC6-1792**	**RC6-2048**	**RC6-2816**
1.01400	771	512	535	534	550	536

With the number of iteration given in 1.01400ns, the amount of total memory consumed by the algorithm in a given time can be computed by multiplying algorithms memory space to the number iteration. The standard RC6 can clearly be observed to have a higher number of iterations within 1.01400ns however; the data it processed is lesser than the modified RC6 Algorithm. For the memory space consumed each algorithm, it was multiplied to the number of iteration. Test shows that the standard RC6 algorithm has a higher memory consumed than the modified RC6 algorithm. This analysis is shown in Table 5. This also shows that the memory requirement of the RC6 algorithm is in Linear Space Complexity with respect to the increasing input to finish a given data.

Table 5. Analysis on Space Consumption

RC6 Algorithm	Iteration	Memory *Iteration(kb/ns)	Bytes of Data	Data consumed	% Processed
RC6 – 128	771	4626	16	12336	1.69%
RC6 - 1024	512	4096	128	65536	8.95%
RC6 – 1280	535	4280	128	68480	9.36%
RC6 – 1792	534	4272	128	68352	9.34%
RC6 – 2048	550	4400	128	70400	9.62%
RC6 – 2816	536	4288	128	68608	9.37%

In figure 9, the graph shows the number of iteration over time where, as the time increases the number of data being processed also increases. Therefore, the RC6 algorithm has a linear time complexity.

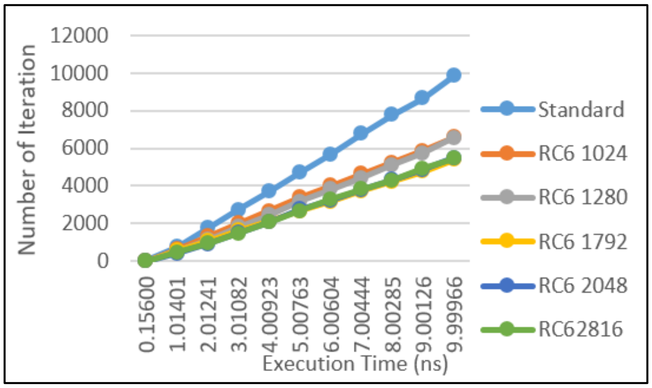

Figure 9. Iteration over Time

5.5 Analysis based on Exhaustive Search

Table 6 shows the test result of the length stored in L array and key generated.

Table 6. The result of Increasing Key Size

Key Size (bit)	Length of L[ ]	Number of Iteration	Key Generated on the Last Iteration
1024	16	132	44
1280	20	132	44
1536	24	132	44
1792	28	132	44
2048	32	132	44
2560	40	132	44
2816	44	132	44

When the key was increased, the number key stored in array L also increases. A 1024 bit key would store 16 keys, 1280 has 20 keys stored, a 1792 bit key would have 28 stored keys, a 2048 bit key would have 32 stored keys, and a 2816 bit key would store 44 keys. This was computed based on Figure 6 Algorithm Step 6.

The table shows that the highest possible keys that can be stored on array L that can be used during the key expansion are a 2816 bit key in 64-bit word size S.

The strength of encryption depends on the key length. Although current encryption algorithms are considered secure, given enough time and computing power as technology advances, they will become vulnerable to brute-force attacks[20]. Thus, the selection of a key in cryptography is very essential as the security of encryption algorithm depends directly on it, the greater the key size the stronger the algorithm will be.

Table 7. Exhaustive Key Search

Key Size	Years to Break
1024	$5.590062111801 \times 10^{271}$
1280	$6.603174603175 \times 10^{348}$
1792	$1.03333333333 \times 10^{503}$
2048	$1.064809806432 \times 10^{580}$
2816	$1.6142784134377 \times 10^{811}$

Table 3 shows the number of years to break a particular key size using mentioned computer specification. A 2816bit key would take 5.02×10^{847}number of the combination of keys and it would take $1.6142784134377 \times 10^{811}$ years to break the key. The computation was done by adopting the formula for exhaustive key search [21].

6. CONCLUSION AND RECOMMENDATION

This paper presented a modification of the RC6 Algorithm using multidimensional key sizes that provide a higher level of security. Increasing the key length would result in a higher number of key stored in L array of the Modified RC6 Algorithm but using a longer key size would result in a much longer time to break the key. This provides an additional level of security on the encrypted message.

The additional register improved the throughput and speed of the algorithm but, on the other hand, the increase in key length had a little effect on the speed of the algorithm during encryption and decryption.

The study was only geared towards evading brute force attack through exhaustive key search and not with other attacks such as differential attack. It is deemed necessary to conduct further research on attacking or breaking the modified algorithm through a differential attack and other attacks to prove the real strength of the modified RC6 algorithm. Performance of the modified RC6 could still be further improved through the enhancement of the word size.

7. ACKNOWLEDGMENT

I would like to express my deepest gratitude to my family, colleagues, and friends for their unending support and valuable suggestions for this paper.

8. REFERENCES

[1] P. Verma, A. Asthana, D. J. Shekhar, and Preety, “A Survey for Performance Analysis Various Cryptography Techniques Digital Contents,” vol. 4, no. 1, pp. 522–531, 2015.

[2] G. Singh and S. Supriya, “A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security,” *Int. J. Comput. Appl.*, vol. 67, no. 19, pp. 33–38, 2013.

[3] M. J. B. Robshaw, “RC6 and the AES,” 2001.

[4] K. Aggarwal, “Comparison of RC6, modified RC6 & enhancement of RC6,” *Conf. Proceeding - 2015 Int. Conf. Adv. Comput. Eng. Appl. ICACEA 2015*, pp. 444–449, 2015.

[5] N. A. El-Fishawy, T. E. El-Danaf, and O. M. A. Zaid, “A modification of RC6TM block cipher algorithm for data security (MRC6),” *Proc. - 2004 Int. Conf. Electr. Electron. Comput. Eng. ICEEC’04*, no. 16, pp. 222–226, 2004.

[6] N. H. Khanapur and A. Patro, “Design and Implementation of Enhanced version of MRC6 algorithm for data security,” *Int. J. Adv. Comput. Res.*, vol. 5, no. 19, pp. 255–232, 2015.

[7] Rajalakshmi and Abarna, “a Modified Approach To Improvise the Efficiency of Des and Aes Using 1024-Bit Key,” *Int. Res. J. Eng. Technol.*, vol. 03, no. 05, pp. 2754–2760, 2016.

[8] G. Ramya and M. M. Anita, “Enhancing Des and Aes With 1024 Bits Key,” *Int. Res. J. Eng. Technol.*, vol. 2, no. 4, pp. 1008–1014, 2015.

[9] B. F. Cruz, K. N. Domingo, F. E. De Guzman, J. B. Cotiangco, and C. B. Hilario, “Expanded 128-bit Data Encryption Standard,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 68, no. 8, pp. 133–142, 2017.

[10] R. Jain, R. Jejurkar, S. Chopade, S. Vaidya, and M. Sanap, "AES Algorithm Using 512 Bit Key Implementation for Secure Communication," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 2, no. 3, pp. 3516–3522, 2014.

[11] K. Kalaiselvi, "Implementation Issues and Analysis of Cryptographic Algorithms based on different Security Parameters," *Int. J. Comput. Appl.*, pp. 23–28, 2015.

[12] R. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 Block Cipher," *First Adv. Encryption …*, 1998.

[13] A. T. Hashim, "A Proposed 512 bits RC6 Encryption Algorithm," vol. 10, no. 1, pp. 11–25, 2010.

[14] K. Aggarwal, "Performance Evaluation of RC6 , Blowfish , DES , IDEA , CAST-128 Block Ciphers," *Int. J. Comput. Appl.*, vol. 68, no. 25, pp. 10–16, 2013.

[15] A. I. Sallam, O. S. Faragallah, and E. S. M. El-Rabie, "HEVC Selective Encryption Using RC6 Block Cipher Technique," *IEEE Trans. Multimed.*, vol. 9210, no. c, 2017.

[16] M. Monger, "RC6: The Simple Cipher," 2004.

[17] H. K. Verma and R. K. Singh, "Enhancement of RC6 block cipher algorithm and comparison with RC5 & RC6," *Proc. 2013 3rd IEEE Int. Adv. Comput. Conf. IACC 2013*, pp. 556–561, 2013.

[18] A. P. Esmita Gupta ME Student, "A new modified RC6 algorithm for cryptographic applications," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 11, pp. 17–23, 2014.

[19] M. Hanzlík, "Space Complexity and Logspace."

[20] W. Yin, J. Indulska, and H. Zhou, "Protecting Private Data by Honey Encryption," *Secur. Commun. Networks*, vol. 2017, 2017.

[21] T. J. Scott, "Cracking Keys with Current Intel CPUs." .