Optimization of CHAM Encryption Algorithm based on Javascript

Chanhui Park[1], Taehwan Park[1], Hwajeong Seo[2], Howon Kim[1]*
[1]School of Electrical and Computer Engineering, Pusan National University, Busan, Republic of Korea
[2]IT Computer System, Hansung University, Seoul, Republic of Korea
* Corresponding Author
{chan70921, pth5804}@pusan.ac.kr , hwajeong@hansung.ac.kr, howonkim@pusan.ac.kr

***Abstract*—** In these days, various web-based services have emerged along with the development of Internet of Things(IoT) and cloud computing technologies. In order to secure security in these IoT and cloud computing environments, Javascript based symmetric key cryptographic algorithms are used. Especially, ARX (Addition, Rotation, eXclusive OR) based block cryptographic algorithms which have higher computational efficiency than SPN(Substitution-Permutation Network) based block cryptographic algorithms are mainly used. In this paper, we implemented the lightweight block cipher CHAM released ICISC 2017 in javascript. IN addition, the computation speed is increased by reducing the number of operations. The proposed technique is experienced in various web browser environments, such as Goggle Chrome, and Mozilla Firefox. It is confirmed that the encryption is faster than the normal algorithm.

Keywords— cryptography; blockcipher; CHAM; javascript;

I. INTRODUCTION

Recently, various web-based services such as Internet and cloud computing have been developed using javascript language [1-5]. In these environments, research is underway on block cipher encryption algorithms implemented in javascript to maintain security and provide reliable services [6,7].

Block cipher algorithms are divided into ARX structure and SPN structure. SPN is a structure that performs encryption by mixing substitute and permutate processes. Substitute is an operation that outputs an input value to another value, and permutate is an operation that complicates input values by changing the array. SPN-based block ciphers include AES [8], PRESENT [9], PRINCE [10], LED [11] and so on. ARX is structure that performs encryption by combining addition, rotation, and eXclusive OR operations. ARX-based block ciphers include KATAN [12], HEIGHT [13], LEA [14], SIMON/SPECK[15] and so on. The ARX requires less computational complexity than SPN and is commonly used in 8 or 16-bit microprocessors that require resource-constrained environments.

In this paper, we discuss optimization and javascript implementation of ARX-based CHAM lightweight cryptographic algorithm released in ICISC 2017.

The rest of this paper is organized as follows: in Section II, we discuss the basic specifications of CHAM and related works. In Section III, we present the novel optimization techniques and javascript implementation. In Section IV, we evaluate the performance of the proposed methods in terms of clock cycles and compare the results with normal algorithm. Finally, Section V concludes this paper.

II. BACKGROUND AND RELATED WORKS

In this section, we describe the CHAM family of lightweight block cipher algorithm and previous research results related to the javascript based block cipher implementation.

CHAM Family of Lightweight Block Ciphers

CHAM family of lightweight block cipher was announced in ICISC 2017[16]. This block cipher is an improvement to the LEA block cipher and is suitable for resource-constrained environment.

The LEA block cipher is known to be a very efficient algorithm on 32-bit ARM processors. However, it is known to be inefficient over 8 and 16-bit. Therefore, NSR developed efficient CHAM block cipher on 8 or 16-bit microcontroller.

CHAM block cipher has an extremely simple key schedule process and round function. For this reason, the CHAM block cipher performs 80 or 96 times round functions to block well known attacks possibilities. Because it used 1 and 8-bit rotation, it is optimized for 8 or 16bit microcontroller.

CHAM block ciphers are divided into CHAM-64/128, CHAM-128/128, and CHAM-128/256 according to block size n and key size k. Table.I is shows the detailed parameters.

TABLE I. LIST OF CHAM BLOCK CIPHER AND THEIR PARAMETERS

cipher	n	k	r	w	k/w
CHAM-64/128	64	128	80	16	8
CHAM-128/128	128	128	80	32	4
CHAM-128/256	128	256	96	32	8

In the Table.I, n is the block size, k is the key size, r is the number of rounds, and w is the word size.

1) Key schedule: Fig.1 is a block diagram showing the key schedule process of the CHAM block cipher.

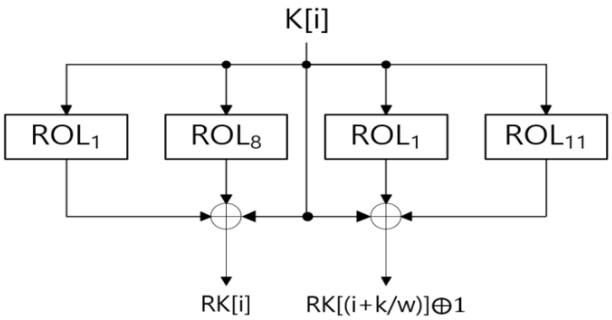

Fig. 1. CHAM Block Cipher Key Schedule

Key schedule process in this block cipher is to generate the round key using only the left rotation and XOR operation from the secret key. This process computes all the round keys from the initial secret key, so the computation process is very simple. In Koo et al. [16], We call this method of key schedule *stateless-on-the-fly*.

The key schedule process can be represented as the following equation.

$$RK[i] = K[i] \oplus ROL_1(K[i]) \oplus ROL_8(K[i])$$
$$RK[i+k/w] = K[i] \oplus ROL_1(K[i]) \oplus ROL_{11}(K[i]) \quad (1)$$

The detailed key schedule algorithm for CHAM-64/128 is Alg. II.1.

Algorithm II.1 KEY SCHEDULE(K)

for $i \leftarrow$ 0 to 7 **do**

RK[i] ←K[i] ⊕ ROL1(K[i]) ⊕ ROL8(K[i])

RK[i+k/w] ← K[i] ⊕ ROL1(K[i]) ⊕ ROL11(K[i])

return (RK)

2) Encryption/Decryption: Fig.2 and Fig.3 are block diagram showing the encryption and decryption process of the CHAM block cipher. According to Fig.2 and Fig.3, this block cipher is 4-branch generalized feistel structure. It encrypts plaintext P = (P[0], P[1], P[2], P[3]), generating a ciphertext C = (C[0], C[1], C[2], C[3]).

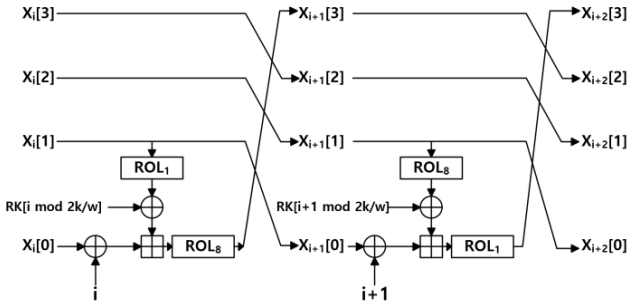

Fig. 2. CHAM Block Cipher Encryption

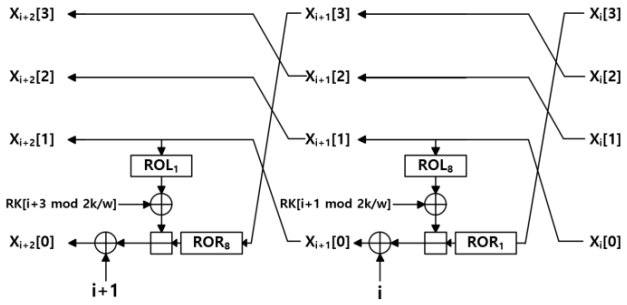

Fig. 3. CHAM Block Cipher Decryption

The CHAM encryption procedure consists of 80 rounds for 64/128 and 128/128, and 96 rounds for 128/256. The encryption equation is as follows.

$$X_{i+1}[3] = ROL_8((X_i[0] \oplus i) \boxplus (ROL_1(X_i[1]) \oplus RK[\, i \bmod 2k/w\,]))$$
$$X_{i+1}[j] = X_i[j+1] \qquad for \quad 0 \le j \le 2 \quad (2)$$

$$X_{i+1}[3] = ROL_1((X_i[0] \oplus i) \boxplus (ROL_8(X_i[1]) \oplus RK[\, i \bmod 2k/w\,]))$$
$$X_{i+1}[j] = X_i[j+1] \qquad for \quad 0 \le j \le 2 \quad (3)$$

Eq. (2) represents the round function when i is an odd number, and Eq. (3) represents the round function when i is even.

The detailed encryption algorithm for CHAM-64/128 is Alg.II.2.

Algorithm II.2 KEY SCHEDULE(K)

for $i \leftarrow$ 0 to 79 **do**

C[0] ← P[1], C[1] ← P[2], C[2] ← P[3]

if i is even **then**

C[3] ← ROL8((P[0] ⊕ i) + (ROL1(P[1]) ⊕ RK[i mod 16]

else

C[3] ← ROL1((P[0] ⊕ i) + (ROL8(P[1]) ⊕ RK[i mod 16]

P[0] ← C[0], P[1] ← C[1], P[2] ← C[2], P[3] ← C[3]

return (C)

We omit the description of the decryption procedure because it is considered the inverse of the encryption procedure.

B. Javascript based Block Ciphers

In Stark et al. [7], they implemented a SPN based symmetric key cryptographic algorithm AES and its operating mode using javascript and evaluated its performance. In this paper, they implemented javascript based AES and performance optimization. Performance evaluation compares performance with other javascript based ciphers in various web browser environments. 585.2kB/s in Chrome, 60.4kB/s in Internet Explorer 8, and 97.4kB/s in firefox, which outperform other javascript based encryption algorithms.

In Seo et al. [6], LEA block ciphers are implemented in javascript and asm javascirpt. The algorithm implemented by the roll and unroll technique was measured in Internet Explorer, Firefox, and Chrome. The measured performance of the

proposed algorithm showed that the performance of javascript improved by 94.3% and 69.8% in chrome and firefox compared to AES. In the case of asm.js, performance was improved by 96.9$$ in chrome, 97.1% in firefox, and 33.3% in Internet Explore.

III. Proposed Methods

In this section, we propose a javascript based optimization methods for CHAM block ciphers.

A. Novel Encryption/Decryption Method

In this paper, we use a method to reduce the number of iterations of the round function to optimize the encryption algorithm. In the case of the existing encryption algorithm, only the first of the four plaintext blocks goes through the round function, and each block moves to the previous block. A total of four rounds is required for all plaintext blocks to go through the round function. Therefore, if all the blocks can go through the round function at once, the number of iterations is reduced by 1/4.

we propose a new method that increases the computation speed by reducing 80 and 96 rounds CHAM block ciphers to 20 and 24 rounds. Fig.4 show a block diagram of the encryption process proposed in this paper.

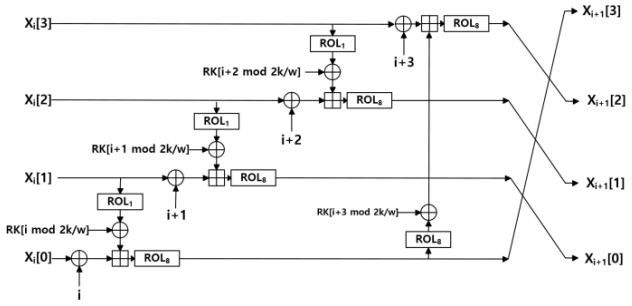

Fig. 4. CHAM Block Cipher Optimized Encryption

In Fig.2, four plaintext blocks are encrypted and intermixed as they go through the round function.

As shown in Fig.4, we propose a method to reduce the number of round function iterations by processing four round functions at once. In the case of decryption, encryption proceeds in the reverse order.

B. Implementation of Javascript based CHAM Block Cipher

For javascript, it supports signed 32-bit integer. Therefore, to implement block cipher based on ARX operation, we use >>> operator to transform to unsigned integer.

1) Addition Operation: The size of the plaintext block is divided into 16 and 32-bits depending on the type of the encryption algorithm. Addition use the + and % operators in the operation.

2) Rotation Operation: When performing a-bit left rotation on operand p in n-bit size, p' = (p ≫ a) >>> 0 | p >>> (n-a) code is used to perform the operation.

In other words, in case of 16-bit plaintext block,

$$P' = (p \gg a) \leftarrow >>> 0 \mid p >>> (16-a)$$

In the case of 32-bit plain text block, the following code is implemented.

$$P' = (p \gg a) >>> 0 \mid p >>> (32-a)$$

3) XOR Operation: For the XOR operation, it can be easily implemented using the ^ operator.

IV. Experiment and Evaluation

In this section, we evaluated the performance of the suggested method on various web browser platform(Firefox and Chrome).

A. Experiment

The performance evaluation of CHAM-64/128, CHAM-128/128, and CHAM-128/256 implemented in this paper was performed on a PC based on Windows. We implemented the proposed algorithm in Firefox and Chrome. Detailed environment is shown in Table.II.

TABLE II. Experiment Environment

Operating System	Windows 10 Ultimate 64bit
CPU	Intel i7-6700 3.40GHz
RAM	16.0 GB
SW	Firefox 56.0
	Chrome 63.0.3239.132

B. Evaluation

The web browsers used in the experiment are firefox 56.0 and Chrome 63.0.3239.132. We calculated the time required for key schedule, encryption, and decryption using the getUTCmilliseconds function in javascript. The performance of the CHAM cryptographic algorithm is in units of cycles/byte, and the performance results are as follows.

Table.III shows the cbp/bytes of the key schedule in firefox and chrome. In the case of the key scheduling process, there is no difference in performance because it is composed of the same algorithm as the existing CHAM block cipher.

TABLE III. Key Schedule Results

Web Browser	**Result (cycles / byte)**
Firefox	22.81
Chrome	13.69

TABLE IV. EXPERIEMENT RESULTS

Cipher	Env.	Method	Enc.	Dec.
CHAM64/128	Chrome	Existing method	438.08	689.07
		Proposed	**296.62**	**588.67**
	Firefox	Existing method	401.57	1245.80
		Proposed	**136.90**	**1213.86**
CHAM128/128	Chrome	Existing method	666.25	1067.83
		Proposed	**479.15**	**926.37**
	Firefox	Existing method	1079.39	949.18
		Proposed	**981.13**	**908.11**
CHAM128/256	Chrome	Existing method	798.59	1282.31
		Proposed	**584.11**	**1154.54**
	Firefox	Existing method	1145.41	1145.41
		Proposed	**1118.03**	**1104.34**

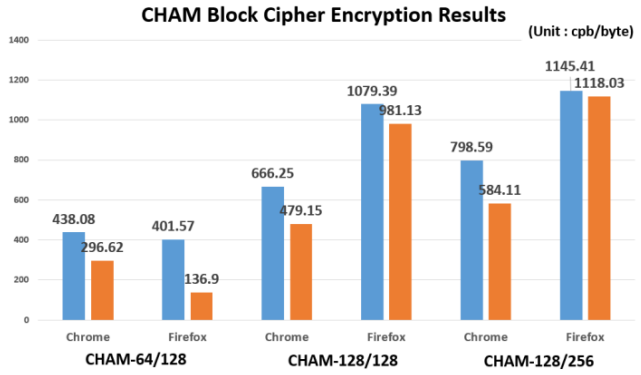

Fig. 5. CHAM Block Cipher Encryption Results

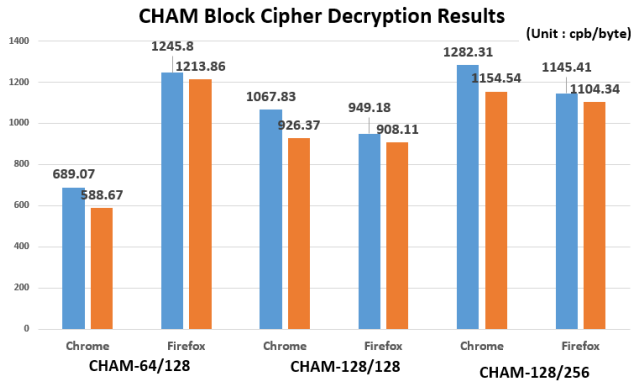

Fig. 6. CHAM Block Cipher Decryption Results

In the Table.IV, the cpb(cycles per byte) decreased over the existing algorithm. Because each web browser has differenct javascript engine, each performance result was measured differently. In particular, cpb decreased significantly in chrome. In the encryption process, it decreased 32.29% in CHAM-64/128, 28.08% in CHAM-128/128, and 26.85% in CHAM-128/256. On the other hand, in the case of firefox, the cpb was decreased in the encryption process of CHAM-64/128, but it was slightly reduced or not significantly different in the other process.

V. CONCLUSION

In this paper, we proposed an optimization method of CHAM block cipher and implemented it using javascript language. We eliminated the unnecessary repetition of the CHAM block cipher to improve the speed of the operation. In addition, we implemented an optimized javascript based cryptographic algorithm for efficient use in IoT and cloud computing environments. The javascript-based CHAM source code can be found in github [1]. Furthermore, performance evaluation is performed on various web browser platforms and compared with the proposed algorithm. As a result, by reducing 80 and 96 rounds to 20 rounds and 24 rounds, it is possible to reduce the number of iterations, respectively.

In this study, we implemented a cryptographic algorithm using pure javascript. In the future, we will be conducting a speed optimization study using asm.js.

ACKNOWLEDGMENT *(Heading 5)*

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.017-0-01791, Development of security technology for energy platform and device)

This work of hwajeong seo was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.NRF-2017R1C1B5075742).

REFERENCES

[1] G. C. Marco. Clipperz online password manager. [Online]. Available: http://www.clipperz.com.

[2] Google browser sync. [Online]. Available: http://www.google.com/tools/firefox/browsersync/.

[3] E. Styere. Javascript aes example. [Online]. Available: http://people.eku.edu/styere/Encrypt/JS-AES.html.

[4] J. Walker. Javascrypt: browser-based cryptography tools. [Online]. Available: http://www.fourmilab.ch/javascrypts/aes.html.

[5] Javascript implementation of aes in counter mode. [Online]. Available:http://www.movable-type.co.uk/scripts/aes.html.

[6] H. Seo and H. Kim, "Low-power encryption algorithm block cipheer in javascript", Journal of information and communication convergence engineering, vol. 12, no. 4, pp. 252-256, 2014.

[7] E. Stark, M. Hamburg, and D. Boneh, "Symmetric cryptography in javascirpt", in Coomputer Security Applications Conference, 2009. ACSAC'09. Annual. IEEE, 2009, pp.373-381.

[8] J. Daemen and V. Rijmen, "Aes proposal: Rijdael", 1999.

[1] https://github.com/ChanhuiPark/CHAM-block-cipher-javascript
https://github.com/ChanhuiPark/cham-block-cipher-js_optimization-

[9] A. Bogtdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, “Present: An ultra-lightweight block cipher”, in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2007, pp.450-466.

[10] J. Borghoff, A.Canteaut, T. Guneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger et al., “Prince-a low-latency block cipher for pervasive computing applications”, International Conference on the Theory and Application of Cryptology and Information Security. Springer, 2012, pp. 208-255

[11] J. Guo, T. Peyrin, and A.Poschmann, “The led block cipher”, cryptographic hardware and embedded system-CHES 2011, 2011

[12] C. De Canniere, O. Dunkelman, and M. Knezevic, “Katan and ktantana family of small and efficienthardware-oriented block ciphers”, Cryptograpic Hardware and Embedded Systems-CHES 2009, Springer, 2009, pp. 272-288

[13] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong et al., “Hight: A new block cipher suitable for low resource device”, International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2006, pp. 46-59.

[14] D. Hong, J.K. Lee, D.C. Kim, D. Kwon, K. H. Ryu, and D.G. Lee, “Lea: A 128-bit block cipher for fast encryption on common processors”, International Workshop on Information Security Applications, Springer, 2013, pp.3-27

[15] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, “The simon and speak lightweight block ciphers”, Design Automation Conference(DAC), 2015 52nd ACM/EDAC/IEEE. IEEE, 2015, pp.1-6

[16] B. Koo, D. Roh, H. Kim, Y. Jung, D.G. Lee, and D. Kwon, “Cham: A family of lightweight block cihers for resource-constrained devices”, International Conference on Information Security and Cryptology(ICISC’17), 2017.