Optimisation of Lightweight Klein Encryption Algorithm With 3 S-box

Seyed Ramin Ghorashi
School of Computing and Mathematics
Charles Sturt University, NSW
Cyber Security Cooperative Research Centre
sghorashi@csu.edu.au

Tanveer Zia
School of Computing and Mathematics
Charles Sturt University, NSW
Cyber Security Cooperative Research Centre
tzia@csu.edu.au

Yinhao Jiang
School of Computing and Mathematics
Charles Sturt University, NSW
Cyber Security Cooperative Research Centre
yjiang@csu.edu.au

***Abstract*— Internet of Things (IoT) have offered great opportunities for the growth of smart objects in the last decade. Smart devices are deployed in many fields such as smart cities, healthcare and agriculture. One of the applications of IoT is Wireless Sensor Networks (WSN) that require inexpensive and space-economic design for remote sensing and communication capabilities. This, unfortunately, lead to their inherent security vulnerabilities. Lightweight cryptography schemes are designed to counter many attacks in low-powered devices such as the IoT and WSN. These schemes can provide support for data encryption and key management while maintaining some level of efficiency. Most of these block ciphers provide good security. However, due to the complex cryptographic scheme's efficiency and optimisation is an issue. In this work, we focus on a new lightweight encryption scheme called the Klein block cipher. The algorithms of Klein block cipher are analysed for performance and security optimisations. A new algorithm which consists of 3-layer substitute box is proposed to reduce the need for resource consumption but maintain the security.**

Keywords*— *Cryptanalysis, Klein encryption, IoT, S-box, WSN.

I. INTRODUCTION

Great attention has been drawn towards the use of Internet of Things (IoT) applications. Industries from healthcare, agriculture and military are using smart objects facilitating their workplaces with the deployment of IoT applications. For example, Wireless Sensor Networks (WSN) and Radio Frequency Identification (RFID) tags are now widely used for object and process tracking in production warehouses and in shipping. A big advantage of the sensors is the low-cost budget required for implementation and maintenance works [1]. The components of WSN device can only offer constrained resources such as limited Central Processing Units (CPU), low memory, and small battery packs. This characteristics of WSN devices has gained much attention regarding their security vulnerability. Thus, maintaining the security level is one of the important tasks for WSN devices.

Due to high demand in the adaptation of IoT devices, the security of many WSN devices has been breached. Implementation of better security leads to other issues because of the requirement of performance and restrictions of resources. Hence, many lightweight cryptography schemes have provided different security approaches for WSN devices with a trade-off between security and performance [2]. Klein encryption is a new lightweight block cipher that has achieved a further step in the trade-off with better software performance on legacy sensor platforms and its compact hardware implementation. Researchers conducted a software performance evaluation on many lightweight block ciphers such as Advanced Encryption Standards (AES) and PRESENT block cipher standards [3]. The evaluation was performed on AVR ATtiny45 microcontroller. The Klein block cipher is part of the evaluation and the results shows promising possibilities. Amongst the results, the Klein block cipher scored average rate of Random Memory Access usage [4] whilst consuming the second lowest energy points [3]. While the Klein block cipher is proved with reliable low-energy resource consumption, the maintenance of security shows vulnerabilities in many cryptanalysis [5]. This suggests the Klein block cipher is prone to security attacks due to weak security implementation. Klein block cipher is optimised for WSN devices which contains less hardware resources. For security vulnerabilities, this becomes trade-off issue between security and performance. The software and security analysis of the Klein block cipher suggests, Klein encryption scheme is invested more on performance and devoted very little on the security aspect [3]. Since this block cipher enjoys the merit of low-resource consumption, the algorithms of the block cipher can be further analysed for security enhancement. In this study, the algorithms of Klein block cipher are analysed against well-known attacks based on performance. With the obtained analysis results, a new algorithm has been designed and proposed to optimise certain parts in the Klein block cipher to consume less resource to implement stronger security standard in place.

II. OVERVIEW OF KLEIN ENCRYTPTION

A. Klein block cipher scheme

The Klein encryption scheme was proposed by Gong et al, in 2011. The Klein block cipher is a family of Substitution-Permutation Network (SPN) with variable key sizes of 64, 80, 96-bits and 12, 16 and 20 iterations respectively [6]. The Klein block cipher has four algorithms per round. *AddRoundKey*, *SubNibble*, *RotateNibble* and *MixNibble*.

a) AddRoundKey: In the first algorithm, the plaintext and the key are combined with exclusive-OR (XOR) operation.

b) SubNibble: The result of XOR'd value is divided into 16 4-bits which are called nibbles. Each nibble are then inserted into the S-box table.

c) RotateNibble: The state is rotated four nibbles (two bytes) to the left per each round.

d) MixNibble: Application of Galois Field, GF(2^8) are applied on two groups of four bytes.

The last algorithm is the adaptation of AES's *MixColumn* algorithm. The involutive 4-bit Substitute-Box (S-box) the

Klein block cipher to operate with less memory requirement [3].

B. Current attacks on Klein encryption

Since the publish of Klein encryption, much cryptanalysis has been discussed including successful attacks in exploiting encryption keys. Two major attack techniques are the iterative differential characteristics analysis and the parallel-cut meet-in-the-middle (PCMITM) attack [7]. Related security analysis observes and shows critical weakness in algorithms of the Klein block cipher.

a) Differential characteristics attack: The differential characteristics attack is well-known cryptanalysis which exploits the keys of the encryption by identifying the input and out differences of a round. One of the techniques that differential characteristics uses is to identify the differences by conditioning the input and out and the path it followed to reach them. Once the differential of six rounds is satisfied, then the exploitation of last *MixNibble* and *RotateNibble* can be determined after the last SubNibble. This is usually at the seventh round which the differences should be zero for higher nibbles. The attack performs on lower nibbles which shifts them through the S-box. The outcome will be reversed through *MixNibble*. Once the results are produced, the attack can be considered feasible if the lower nibbles are active. To reduce the key recovery costs, the trials can be reduced 258 times which the attacks always work within the number of trials [9-12].

b) Parallel-cut meet-in-the-middle attack: The PCMITM attack application to Klein block cipher could have potential success if *AddRoundKey*, *SubNibble* and *RotateNibble* have diffusion between the higher and lower nibbles. In addition, the PCMITM attack requires the condition of a nibble separation. In the *MixNibble* algorithm, the result of higher and lower nibbles produce a diffusion rate. Therefore, for PCMITM to be performed, it must compute the higher and lower nibbles [5-8].

III. OPTIMISATION APPROACHES

The Klein block cipher is designed for WSN and IoT devices which are limited with hardware resources. To fit into these devices, a trade-off issue between security and performance is adopted [3, 8-10].

Fast encryption system depends on its encryption key, the designed cipher block size, and the plaintext. To speed up the encryption process, a common design is to reduce the produced computational overload whilst trying to maintain the security level consistent.

A. Optimisation of alternative algorithm

The four algorithms in the Klein block cipher have different functions and consume resources differently. The overall software performance of Klein-64 has been measured and evaluated before [11]. However, the individual performance of the algorithms in the Klein block cipher has not been evaluated and remains unknown. We first analyse and evaluate the performance of each algorithm in Klein encryption. Secondly, we propose an alternative algorithm that replaces the fourth algorithm in the Klein block cipher. Evaluation of original and alternative algorithms will be conducted to measure and compare the resources usage such as CPU percentage, memory bytes, execution time and the data processed. These components are the most important sources of measurement in the software analysis. These components determine the usage of each algorithm by providing logical data. The WSN devices are small computational units which contain only a limited amount of these components [12]. Hence it is important to understand the amount of resource is used for each algorithm.

B. Optimisation experiments

Since the reference code of the Klein block cipher original paper was not published, we an implementation of Klein-64 has been written in Python 3 programming language. To make sure that the integrity and the consistency of the application have met, the design of the program has been followed from the published article [6]. For embedded systems such as microcontrollers, many programming languages are used to program the devices. These programming languages include JavaScript, C and Python. In this study, it is intended to use Python language because of the open-source license and wide range of libraries available across the Python community.

For these experiments, we have chosen to implement the Klein program on two devices. The first device was a Windows computer that has CPU clock speed of 2.4 GHz and 8 GB of Memory. The second device was a Raspberry Pi 3 that has 1.5 GHz of CPU clock speed and 1 GB of Memory. The experiments ran on Raspberry Pi 3 to simulate the implementation on a WSN device.

Microcontrollers are also a family of WSN which are made of small computers with limited resources. These devices are responsible of controlling other sensors. For further analysis of this study, the modified algorithms are considered to be implemented on these devices [13].

The execution of each algorithm has been conducted once in a Windows computer and once in the Raspberry Pi 3 device. The implementation was done by acquiring four different programs with consistent information given to evaluate the data, CPU speed, memory usage and time each program took [14].

The implementation of 3 layers of S-box was conducted after the implementation of original algorithms. The 3 layers of S-box is a 4-bit permutation that applies to 16 nibbles after *RotateNibble* algorithm. For this algorithm, to ensure the message's integrity, the state is XOR'd with the key once before entering the 3 S-box algorithm and once in between of each S-box. Table 1 represents the values of 3 S-box.

TABLE I
3 S-BOX USED IN LAST ALGORITHM OF KLEIN BLOCK CIPHER

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	B	F	8	C	9	E	7	6	2	4	D	0	3	A	5	1
$(x)^{1+x}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	8	3	F	1	6	B	4	E	0	C	D	5	9	A	7	2
$(x)^{2+x}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	9	B	4	7	2	C	E	3	F	0	D	1	5	A	6	8

The execution of the original algorithms was experimented on two different devices to evaluate the effect

of software performance. The first experiment was on a Windows computer. The result of *AddRoundKey* had higher percentage rate than the *SubNibble* and *RotateNibble*, however, in *MixNibble*, a bigger percentage rate was shown. The usage of the memory for all programs have shown similar result but in terms of the execution time, *AddRoundKey* and *MixNibble* had longer time taken compared to *SubNibble* and *RotateNibble*. The results is shown in Table 2.

TABLE II
THE EVALUATION OF ORIGINAL ALGORITHMS OF KLEIN BLOCK CIPHER IN WINDOWS COMPUTER

Algorithm	CPU (%)	RAM (KiB)	Time (Seconds)	Data (Byte)
AddRoundKey	35.0	8500	0.078095	289
SubNibble	25.0	8500	0.015622	89
RotateNibble	25.0	8468	0.015626	178
MixNibble	58.3	8624	0.0781302	264

The results of algorithms executed on Raspberry Pi 3 showed a different percentage of CPU and Memory. In this implementation, the CPU percentage of *AddRoundKey* and *SubNibble* were reasonably low compared to *RotateNibble,* however, *MixNibble* had the highest percentage of all. The results of the memory percentage increased towards the last algorithm. However, the time used for each algorithm to execute was different. The execution time of *SubNibble* and *RotateNibble* were exactly the same. However, the execution time of *MixNibble* was the highest amongst other algorithms. The results of the original algorithms on Raspberry Pi device is shown in Table 3.

TABLE III
THE EVALUATION OF ORIGINAL ALGORITHMS OF KLEIN BLOCK CIPHER IN RASPBERRY PI 3

Algorithm	CPU (%)	Memory (%)	RAM (KiB)	Time (Seconds)	Data (Byte)
AddRoundKey	7.0	1.8	17268	0.141437	289
SubNibble	7.0	1.8	17292	0.065789	89
RotateNibble	9.7	1.8	17244	0.065367	178
MixNibble	16.0	1.8	17324	0.260921	264

The implementation of 3 S-box is a replacement for MixNibble algorithm. Thus, the same amount of data has been used to execute the algorithm in both devices. The result for 3 S-box in Windows device showed higher CPU rating when executed, the memory usage was at 8352 bytes, the duration of execution took almost 0.05 seconds and the data processed was 267 bytes. The results is shown in Table 4.

TABLE IV
THE EVALUATION OF 3 S-BOX IN WINDOWS COMPUTER

Algorithm	CPU (%)	RAM (KiB)	Time (Seconds)	Data (Byte)
3 S-box	25.0	8352	0.0468690	267

The result for 3 S-box in Raspberry Pi showed high CPU percentage when conducted, the memory usage was at 17076 bytes, the duration of the execution took almost 0.09 seconds and the data processed was 267 bytes. Table 5 shows the results.

TABLE V
THE EVALUATION OF 3 S-BOX IN RASPBERRY PI 3

Algorithm	CPU (%)	Memory (%)	RAM (KiB)	Time (Seconds)	Data (Byte)
3 S-box	27.0	1.8	17076	0.088864	267

IV. DISCUSSION AND FUTURE WORK

The two approaches were performed on Klein encryption with 64-bit key size. The experiment of these approaches was conducted on two different devices to simulate the results. The first experiment consisted of analysing the software performance of the original algorithms of Klein block cipher. Although the result for each device had different percentages, each algorithm produced similar results when executed on both devices.

In terms of the high CPU percentage of *AddRoundKey* performed in Windows computer is due to the operation that is handled. The XOR gate can have higher transistor than other logical gates such as AND gate and OR gate [15]. A similar effect has been recorded in *MixNibble* algorithm. Since *MixNibble* applies the Rindael's *MixColumn* from AES, it multiplies two columns together and performs XOR operation between each column [16, 17]. The difference of CPU percentage of *AddRoundKey* and *MixNibble* is 49% increase. A small increase of 1.45% of memory usage is shown between the first and last algorithm. However, the time of the execution of these algorithms are exactly the same with 0.078 seconds. This shows that with the same amount of time a higher CPU resource has been consumed.

In Raspberry Pi 3 device, the same experiment performed to analyse the performance speed of each algorithm. The records show similar results in Raspberry Pi except for the CPU usage for *AddRoundKey* was not overloaded as it did in Windows computer. This could be due to the resource management of each device. Nevertheless, *MixNibble* was recorded with the highest CPU consumption out of other algorithms. The execution time of *AddRoundKey* was higher than *SubNibble* and *RotateNibble*. The reason the first algorithm had higher execution time than second and third algorithms are due to the data size being processed and the operation of the algorithm itself. *SubNibble* and *RotateNibble* are performing bit-wise operations while *AddRoundKey* is performing XOR operation.

The second approach analysed the effect of 3 layers of S-box based on software performance. This approach was conducted on both Windows computer and Raspberry Pi 3 device. The result of 3 S-box on these devices showed high execution of CPU, this was because the program had 3 XOR operations plus 3 sets of *SubNibble* algorithms. The operation does consume higher resource at the initial point however, the consumption of memory for both devices were less than the results recorded for *MixNibble* algorithm. The time that each took for execution were 0.05 and 0.09 seconds respectively. When 3 S-box (in Windows computer) is compared with the *MixNibble* algorithm ran in Windows computer, *MixNibble* had produced as twice as much CPU. The amount of memory *MixNibble* used was 8624 Kilobytes and the amount of memory 3 S-

box used was 8352 Kilobytes. A difference of 272 Kilobytes. In terms of the time difference, there was a delay in *MixNibble* algorithm of 0.0313 seconds.

In Raspberry Pi device, *MixNibble* algorithm was executed 1.68% faster than 3 S-box. The memory usage for 3 S-box was achieved to execute with 248 Kilobytes faster than *MixNibble*. Also, 3 S-box was able to achieve to execute within 0.0172 seconds faster than *MixNibble* did.

The results recorded showed that 3-Sbox is not efficient for CPU's as they would have to run at higher rates for full rounds, however, in terms of memory usage, time efficiency and implementation of cost, 3-Sbox has shown better results.

In terms of the security level, 3 S-box provides some security protection against side-channel attacks. This is done by implementing 3 layers of S-box for bit-wise operation. This is also provided by 3 XOR operations in between each S-box.

Previous results have evaluated many lightweight conventional cryptography schemes based on the performance of the scheme and the memory consumption [6]. These evaluations show the effect of software performance for each block cipher for full iterations. However, with the consideration of our study, the software performance of Klein block cipher algorithms has not been conducted. With close analysis of the trade-off between performance and security, this study identifies the resource consumption of each algorithm. Despite the security strength of Klein block cipher, the performance of the Klein block cipher with the new algorithm is improved. Reducing the weight of the performance off WSN and IoT devices improves the efficiency of the software. In other words, it has less performance stress on the physical device and allows more resources available for security implementation.

In future work, we plan to apply this experiment on new approaches for increased security and less computational overload. The two new approaches include analysing the effect of logical gates in *AddRoundKey*, 3 S-box and *MixNibble* algorithms for logical gates with less rate of diffusion. Logical gates such as AND gate and OR gates.

For future work, the Klein block cipher is considered to be programmed in C programming language. This language is considered to have faster Read/Write to memory than other high-end programming languages which causes for a faster and more responsive software execution.

The implementation of these gates on 3 S-box could reduce the CPU and memory consumption by reducing the number of XOR gates [15].

The other approach is to increase the iteration of Klein block cipher for enhanced security. The authors of [18] shows the attack of key recovery on Klein-64 within 8 rounds out of 12. With increased iterations of Klein block cipher and implementation of 3 S-box, it is possible to overcome the security issue whilst keeping the memory consumption low.

ACKNOWLEDGMENT

The work has been supported by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme.

REFERENCES

[1] S. Oza and D. Mathpal, "A Study on Internet of Things Security and Lightweight Cryptography", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN: 2456-3307, Volume 3, Issue 3, pp.683-689, 2018.

[2] S. F. S. Adnan, M. A. M. Isa, and H. Hashim, "Energy analysis of the AA β lightweight asymmetric encryption scheme on an embedded device," in *2016 IEEE Industrial Electronics and Applications Conference (IEACon)*, 2016, pp. 116-122: IEEE.

[3] T. Eisenbarth, Z. Gong, T. Güneysu, S. Heyse, S. Indesteege, S. Kerckhof, F. Koeune, T. Nad, T. Plos, F. Regazzoni, FX. Standaert and L. van Oldeneel tot Oldenzeel, "Compact implementation and performance evaluation of block ciphers in ATtiny devices," in *International Conference on Cryptology in Africa*, 2012, pp. 172-187: Springer.

[4] T. M. Fernández-Caramés and P. Fraga-Lamas, "A Review on the Use of Blockchain for the Internet of Things," *IEEE Access,* vol. 6, pp. 32979-33001, 2018.

[5] A. Aghaie, M. M. Kermani, and R. Azarderakhsh, "Reliable and Fault Diagnosis Architectures for Hardware and Software-Efficient Block Cipher KLEIN Benchmarked on FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 37, no. 4, pp. 901-905, 2017.

[6] Z. Gong, S. Nikova, and Y. W. Law, "KLEIN: a new family of lightweight block ciphers," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 2011, pp. 1-18: Springer.

[7] I. Nikolić, L. Wang, and S. Wu, "The parallel-cut meet-in-the-middle attack," *Cryptography and Communications,* vol. 7, no. 3, pp. 331-345, 2015.

[8] N. Ferrari, T. Gebremichael, U. Jennehag, and M. Gidlund, "Lightweight Group-Key Establishment Protocol for IoT Devices: Implementation and Performance Analyses," in *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, 2018, pp.31- 37: IEEE.

[9] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE transactions on Information Theory,* vol. 26, no. 4, pp. 401-406, 1980.

[10] D. Irwin, P. Liu, S. Chaudhry, M. Collier, and X. Wang, "A Performance Comparison of the PRESENT Lightweight Cryptography Algorithm on Different Hardware Platforms," in *2018 29th Irish Signals and Systems Conference (ISSC)*, 2018, pp. 1-5: IEEE.

[11] M. Cazorla, K. Marquet, and M. Minier, "Survey and benchmark of lightweight block ciphers for wireless sensor networks," in *2013 International Conference on Security and Cryptography (SECRYPT)*, 2013, pp. 1-6: IEEE.

[12] R. Garg, A. L. Varna, and M. Wu, "An efficient gradient descent approach to secure localization in resource constrained wireless sensor networks," *IEEE Transactions on Information Forensics and Security,* vol. 7, no. 2, pp. 717-730, 2012.

[13] H. Guo, K.-S. Low, and H.-A. Nguyen, "Optimizing the localization of a wireless sensor network in real time based on a low-cost microcontroller," *IEEE transactions on industrial electronics,* vol. 58, no. 3, pp. 741-749, 2009.

[14] J. Grossschadl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed, "Energy evaluation of software implementations of block ciphers under memory constraints," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1-6: IEEE.

[15] D. Canright, "A very compact S-box for AES," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2005, pp. 441- 455: Springer.

[16] V. Lallemand and M. Naya-Plasencia, "Cryptanalysis of KLEIN," in *International Workshop on Fast Software Encryption*, 2014, pp. 451-470: Springer.

[17] C. Nandini and K. Rekha, "A Review of AES and Visual Cryptographic techniques for added security," *Perspectives in Communication, Embedded-systems and Signal-processing- PiCES,* vol. 1, no. 2, pp. 6-7, 2017.

[18] J.-P. Aumasson, M. Naya-Plasencia, and M.-J. O. Saarinen, "Practical attack on 8 rounds of the lightweight block cipher KLEIN," in *International Conference on Cryptology in India*, 2011, pp. 134-145: Springer.