Performance Analysis of CLEFIA, RC5 ,BlowFish,SEAL,and GOST Ciphers For Implementation of Secure RFID System

# SanjeevKumar , Dr. Haider Banka, Dr B Kaushik

[*sanjeevnarg@gmail.com*,](mailto:sanjeevnarg@gmail.com,%20) *haider.banka@gmail.com bkaushik99@gmail.com*

***Abstract -* An efficient light weight block cipher is necessary to achieve end-to-end security with the increase growth of IOT. So now-a-day Light weight cryptography is a recent research topic among researchers by preserving data integrity and security. In this paper we propose the performance analysis of five light weight protocols-SEAL, BLOWFISH, CLEFIA,RC5, GOST. Analysis of these protocols are evaluated by the parameters like throughput, encryption time, execution time ,decryption time and execution time. All of these parameter are evaluated on tested on different Plaintext Key sizes 16bits,32bits,64bits,128bits,256bits,512bits,1024bits,2048bits.A comparative analysis is performed by analysis of result based on these parameters**

**Keywords –** IoT Security, Lightweight Cryptography, STM32F401 MCU.

1. INTRODUCTION
2. **Distributed Architecture of Secure RFID System**

Tag (Transponder)

I

N

T

E

R

R

O

G

A

T

I

O

N

R

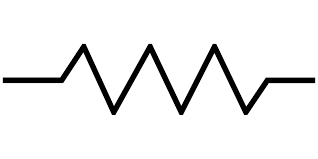
E

P

L

Y

Inductive Coupling



Reader

(Tranciever)

Interface cabling

Interface Unit

(MicroProcessor/Micro Controller Based System)

Application Program

(Encryption Mechanism)

Captive Network

Authentication Server

Figure :2

## II. Performance Evaluation Schema.

## The performance evaluation of five different leight weight encryption algorithm is formed as per the schema depicted in figure 2.All five algorithm are evaluated on three distinct criterion viz decryption time, percentage CPU utilization and memory usage(in %).A plain text generator function described as f(k)=22K bytes, where k is 1-4 is used to provide feed to all five algorithm. This means all the five algorithm need to encrypt 16,32,64,128 bytes plaintext to generate cipher text asper their internal process of S-box key scheduling, pseudorandom function etc. An intermediate input file of cipher text is generated called ciph.dat. that contains cipher data algorithm. The computation of total cpu time utilized in decryption of each 22k bytes of plain text is achieved using chrono class of timer for cpu and subtracting the instant time

Input File

Decryption Time(32 bytes)

Decryption Time(16 bytes)

Decryption Time(2048 bytes) bytes)

C

L

E

F

I

A

Decryption Time(2048 bytes) bytes)

Decryption Time(32 bytes)

Decryption Time(16 bytes)

Decryption Time(16 bytes)

Decryption Time(32 bytes)

Decryption Time(2048 bytes) bytes)

Clefia

Cipher data

1 second cyclic timer

Generator function = 2

Seal

D

E

C

R

Y

P

T

I

O

N

Clefia

Gost

BlowFishh

RC5

**GUI (Plots& Textual Display)**

Average Decryption Time=1/10∑Nb where b:1-4 and Nb=Generator Function output

B

L

O

W

F

I

S

h

Decryption Time(16 bytes)

Decryption Time(2048) bytes)

Decryption Time(32 bytes)

Decryption Time(16 bytes)

Decryption Time(2048) bytes)

Decryption Time(32 bytes)

GOST

RC5

S

E

A

L

E

N

C

R

Y

P

T

I

O

N

Gost

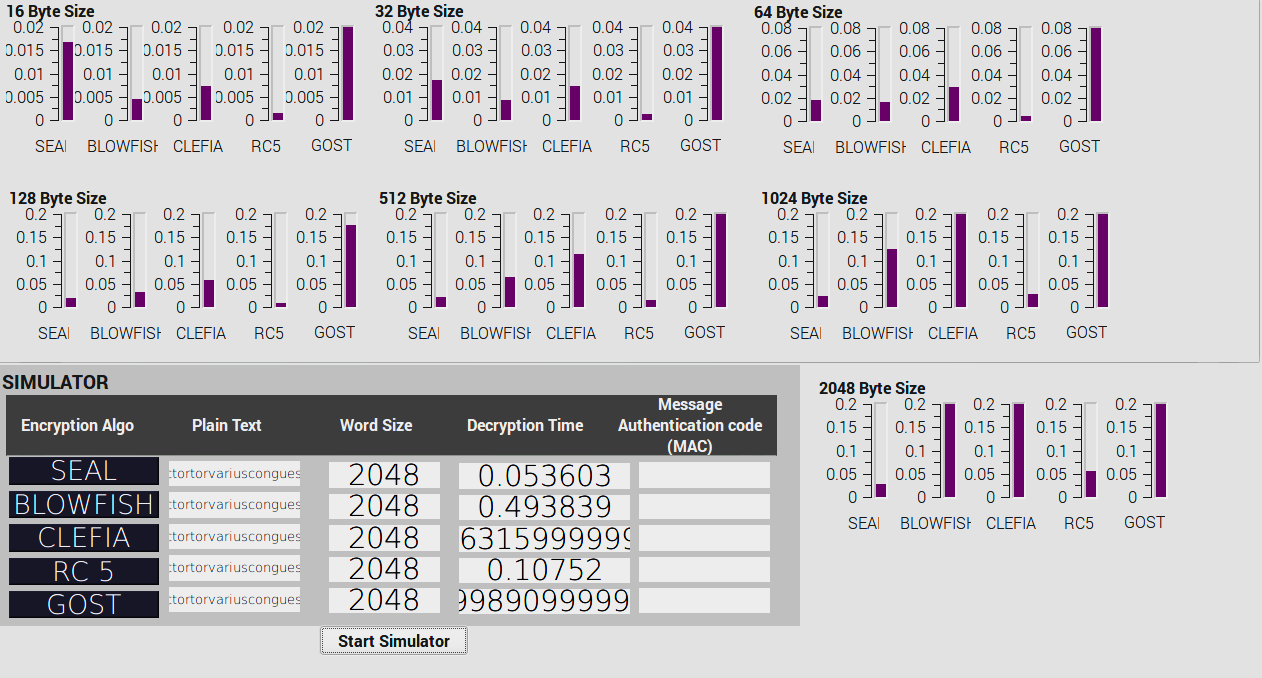
BlowFish

RC5

Seal

III The Data simulator Application

The performance evaluation of light weight encryption algorithm is achieved by implementing a simulator application. The simulator is a GUI based application program implement in QT with some widget such as push button ,bar graph, text editor etc. The GUI of the simulator used in evaluation of the algorithm is shown in figure 3.It uses a cyclic timer class that is used to stream the cipher text data to each algorithm and is activated using a small pushbutton” simulator”. The time used is decryption is depicted graphically and plain text stream with byte size and corresponding decryption time is depicted in textual mode through text editor widget.

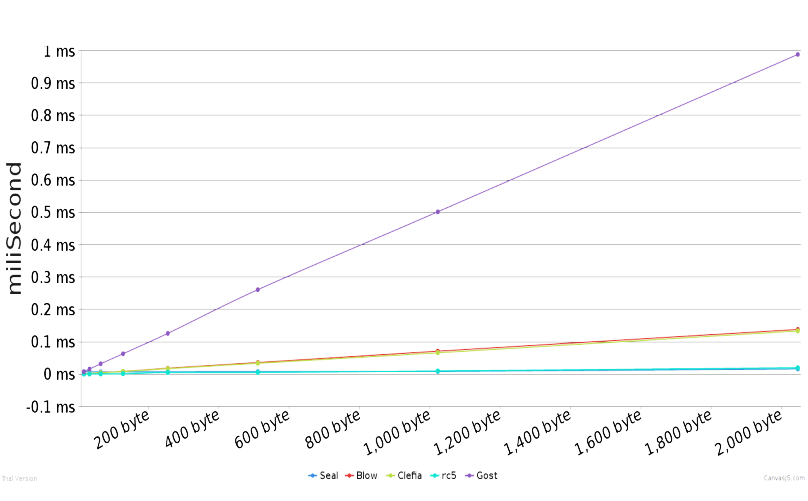
****

The performance evaluation of light weight encryption algorithm is achieved by implementing a simulator application. The simulator is a GUI based application program implement in QT with some widget such as push button ,bar graph, text editor etc. The GUI of the simulator used in evaluation of the algorithm is shown in figure 3.It uses a cyclic timer class that is used to stream the cipher text data to each algorithm and is activated using a small pushbutton” simulator”. The time used is decryption is depicted graphically and plain text stream with byte size and corresponding decryption time is depicted in textual mode through text editor widge

1. **Methodology of Performance Analysis**

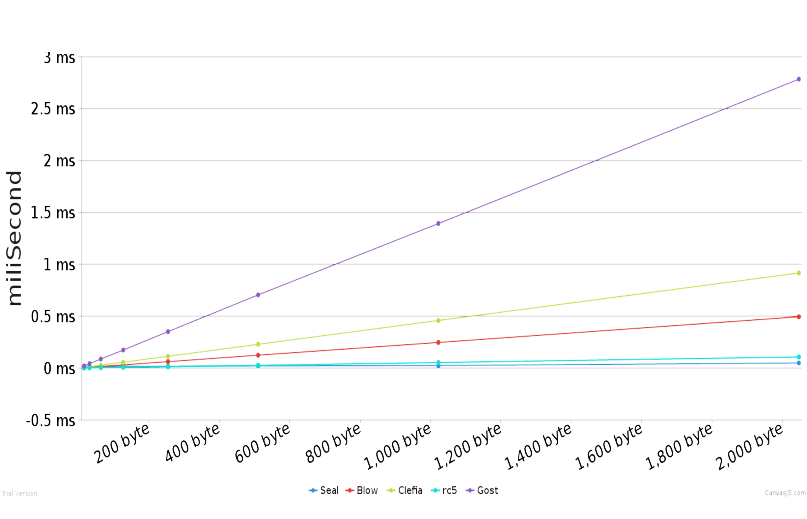
In order to evaluate the performance of the encryption algorithms two basic category of criteria was taken into account in our research work:-first one is total decryption time of the algorithms and other resource usage. The rational of taking into account of taking two different criteria as mentioned above is first the response time of the secured system should be minimal if it is trying to decrypt the data from the tag and other resource usage because RFID systems are by default memory and processing capability constrained devices. If the resource usage of the implemented system will be more than power requirement shall increase as most of the implementation in such IOT environment is constrained by low power requirement.

In order to evaluate and justify the selection of a particular light weight algorithm for implementation of the proposed secured RFID system a simulation run was carried out on two architecturally different systems. One of the system is a virtual machine based system which is fundamentally a X86 system on ubuntu OS and other is ARM processor based system (the actual target system) on sane ubuntu variation of OS.

1. **Performance on System 1**

The details of system 1 which is a X86 based virtual machine is specified in table 2 below.Simulation run for system 1 was carried out as per schema described in section II , the graph of the decryption time versus the number of plain text byte size is shown in figure 3

|  |  |
| --- | --- |
| **System Aspect** | **Description** |
| **Architecture:** | X86\_64 |
| **Byte Order** | Little Endian |
| **CPU cores** | 2 |
| **OS** | Ubuntu Kernel version |
| **Processor Class/Type** | Intel 64/32 bit operation mode |
| **CPU Max Frequency** | 2594.112 MHz |
| **CPU Min Frequency** | 600 |

****

1. **Performance on System 1**

The details of system 2 which is a ARM processor based v machine is specified in table 3 below.Simulation run for system 2 was carried out as per schema described in section II , the graph of the decryption time versus the number of plain text byte size is shown in figure 4

|  |  |
| --- | --- |
| **System Aspect** | **Description** |
| **Architecture:** | armv7l\_32 |
| **Byte Order** | Little Endian |
| **CPU cores** | 4 |
| **OS** | Ubuntu Kernel version |
| **Processor Class/Type** | ARMv7 Processor rev 4 (v7l) |
| **CPU Max Frequency** | 1200 MHz |
| **CPU Min Frequency** | 600 |

**CLEFIA** supports 128-bit block size with three different key sizes: 128-bit, 192-bit, 256-bits. Structure of CLEFIA is as shown in Figure 2. This algorithm is an ISO/IEC 29192-2 standard lightweight crypto cipher [10] currently available. The basic building block of this algorithm is the GFN (d, r) where d denotes the data branch and r is round. Data processing part of the CLEFIA takes four 32-bit whitening key 2r 32-bit round key and 128-bit plain text for encryption. The two F-functions are used which are simple substitution and permutation (4x4 diffusion matrix).

Key scheduling part takes input key to derive the intermediate key. CLEFIA128 uses GFN(4,12) and 60 32-bit constants, CLEFIA192 uses GFN(8,10) and 84 32-bit constants, and CLEFIA256 uses GFN(8,10) and 92 32-bit constants respectively to generate intermediate key from the input key. These intermediate keys are updated every two rounds with the DoubleSwap function. Intermediate keys are expanded with input key to derive four whitening keys and 2r round keys[13].

*Figure 2: Structure of CLEFIA [13].*

Round-12,13,14 are susceptible to integral attacks whereas round-13,14,15 suffer from improbable differential attacks [11] [12]. So, round-18,22,26 are preferable for 128,192,256-bit key sizes to provide security against these attacks. Two S-Boxes are used to overcome the byte ordering saturation attack and algebraic attacks including XSL attack. Two different diffusio

**Main Thread (Front End)**

GUI

Write Signal

Data Input

Unsigned long buffer conversion

Cipher Text to Hex format

Read/write Process Activation

Pipe

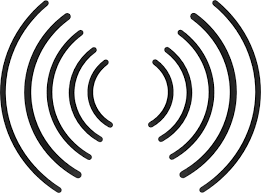
RF link configurations

**Read Write Thread (Back End)**

**T**

**A**

**G**



matrices are used to provide immunity against differential and linear attack [13].

**TWINE** uses 64-bit block size and supports two key sizes: 80-bit and 128-bits. In the data processing part of the algorithm, 64-bit plain text, 36 32-bit round keys are

taken to provide a 64- bit cipher text. Round function of TWINE is very simple (Figure

3) where in each round, eight F-functions are called which does simple XORing plaintext with sub key and applying 4x4 S-Box. Permutation (n) uses a more sophisticated approach to speed-up diffusion compared to CLEFIA which does simple circular shift. Here, only half of the circular shift rounds are required to diffuse to all sub blocks. Decryption in TWINE uses same S-Box, key schedule as encryption but the diffusion layer considered is the inverse of encryption [4].

In key scheduling part of TWINE, input key uses 35 6-bit constants to produce a 36 32-bit round keys. Key schedule of TWINE provides an on-the-fly operation of producing round key by sequentially updating its key state. By doing this hardware footprint reduces which conversely increase the performance. Since the round keys are updated sequentially there is no need for bit permutation or the intermediate key generation[9].

Full cipher TWINE80 and TWINE128 are susceptible to biclique attacks [8]. 23-round TWINE80 and 25-round TWINE128 suffer from zero-correlation attacks. Therefore, 36 rounds for both the key sizes are recommended which gives acceptable security enhancement [9] [20].

*Figure 3: Round function of TWINE [4].*

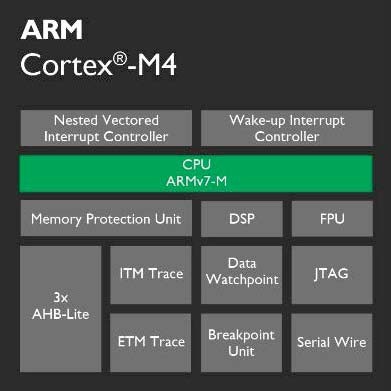
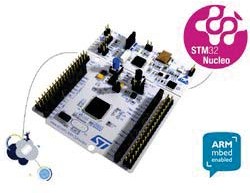
## LIGHTWEIGHT BLOCK CIPHER IMPLEMENTATION ON STM32FMCU

This section provides information on how the mentioned ciphers were implemented on STM32 (ARM Cortex M4) microcontrollers. A brief introduction regarding the platform selected, how porting of the cipher is done and the software development tool used is given next.

The platform specification considered for this project is ARM Cortex M4 which are a family of 32-bit RISC MCUs and uses ARMv7E-M architecture with 3 stage pipelining which result in an ideal average CPI (clocks per Instruction) of1.67

[21] [22]. Due to its high-energy efficiency (with low dynamic power and integrated software controlled sleep modes), performance, and inbuilt powerful trace technologies, ARM Cortex-M4 microcontrollers have reached a high popularity in

cost sensitive embedded device requiring minimal area configuration [23].



*Figure 4: STM32F4 MCU and ARM Cortex-M4[23].*

Since the aim of this project is to implement lightweight block ciphers in IoT environments, STM32F401RE is considered as the target platform, which is specifically designed for these environments [24] [25] and supports the specifications mentioned in Table 1[25].

*Table 1: STM32F401RE Hardware Specification.*

|  |  |
| --- | --- |
| **Core** | **ARM 32 Cortex M4** |
| **CPU Frequency** | 84 MHz (84,000,000 cycles per  sec) |
| **Flash Memory** | 512 KBytes |
| **SRAM** | 96Kbytes |
| **Security** | MPU (Memory Protection Unit) |
| **USB Type** | USB OTG FS |
| **Supply Voltage (V) max** | 3.6 |
| **Supply current (per MHz)** | 137 (μA ) |

Implementations of the crypto algorithms which were written in C language were referenced [29][30], and were initially ported and tested on the MCU. These tests were not successful as the referenced codes were found incompatible with our MCU’s firmware framework. These referenced codes were redesigned using embedded C++ language to shorten the size of the code, to achieve maximum performance and to implement the crypto algorithms (CLEFIA, PICCOLO, TWINE) on our MCU’s hardware. The code was also modified to analyze the performance metrics like memory efficiency, energy consumption, execution time, and throughput of the crypto cipher algorithms running on the MCU. Embedded C++ language is selected to implement the code among different available programming languages like Assembly, java and others, as it provides many appealing functionalities and characteristics.

One of the most important characteristics of Embedded C++ language is that it is similar to the assembly language in terms of performance and code size. It is an efficient, fast and highly portable language [16] which is also easy to build and debug

[18] [14]. Lightweight block cipher specifications with their block size and key sizes are shown in Table 2.

Online mbed compiler and Keil uVision5 is used as the IDE to implement ciphers on STM32F4 MCU. They are open source

and provides rich features of MCU environments. Online mbed compiler makes the coding portable and compiles the source code directly into binary files which can then be flashed directly on to the MCU by just click and drag [19]. Once the flash succeeds, the MCU flash drive will reload and the green light will be turned ON in the MCU. Keil uVision5 software is supported in Windows and provides a sophisticated full-scale debugger which can be used to monitor the serial port, perform data tracing andetc.

*Table 2: Lightweight Block Cipher Specifications.*

|  |  |  |  |
| --- | --- | --- | --- |
| **KeySize BlockSize # Rounds** | | | |
| **CLEFIA** | 128 Bit | 128-Bit | 18 |
| 192 Bit | 128-Bit | 22 |
| 256 Bit | 128-Bit | 26 |
| **PICCOLO** | 80 Bit | 64-Bit | 25 |
| 128 Bit | 64-Bit | 31 |
| **TWINE** | 80 Bit | 64-Bit | 36 |
| 128 Bit | 64-Bit | 36 |

## PERFORMANCE ANALYSIS &COMPARISION

In this part of the section, we discuss a set of benchmark parameters like execution time, throughput of the encryption process, memory consumption, energy consumption which are used to calculate and determine which algorithm is best suited for IoT environments. We also provide information regarding how these are measured with respect to the STM32F4MCU.

Speed of the operation or the execution time is one of the key metric used to evaluate the performance of the block ciphers [27]. For this, timer function is used (provided by the mbed package) which is executed before the encryption operation and stopped after it finishes. In our analysis, for each block as the plaintext size increases, key schedule is also performed and included in the execution time. The results are tabulated and analyzed asfollows.



**CLEFIA EXECUTION TIME**

120

100

80

60

40

20

128 BitKey

192 BitKey

256 BitKey

EXECUTION TIME (MS)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 512 | 1024 | 2048 | 3072 |
|  |  |  |  |  |
|  |  |  |  |  |
|  | 12.329 | 24.782 | 49.105 | 73.863 |
|  |  |  |  |  |
|  |  |  |  |  |
|  | 16.934 | 33.521 | 67.76 | 100.893 |
|  |  |  |  |  |
|  | 18.562 | 36.89 | 73.727 | 110.635 |
|  |  |  |  |  |
| PLAINTEXT SIZE (BYTES) | | | | |

*Figure 5: CLEFIA encryption execution time.*

CLEFIA is executed for 128 bit, 192 bit, and 256 bit Keys for plaintexts of size 512 Bytes, 1024 Bytes, 2048 Bytes, and 3072 Bytes and the average values were calculated and tabulated as shown in Figure 5. As we see, the encryption is increasing exponentially as the plaintext size increases. This is because the block cipher uses two feistal function and two diffusion matrixes. While the decryption follows a similar procedure with only changes made to the order of round keys and whitening keysselection.



**PICCOLO EXECUTION TIME**

400

350

300

250

200

150

100

50

0

512 1024 2048 3072

80BitKey 45.32 90.517 180.865 217.512

128BitKey 57.662 115.291 230.445 345.618

PLAINTEXT SIZE (BYTES)

EXECUTION TIME (MS)

*Figure 6: PICCOLO encryption execution time*.

In PICCOLO, the operational time for each key size and plaintext sizes are shown in Figure 6. Here, we observe the exponential raise of encryption time as the plaintext increases but when we compare two key sizes we can see more growth in 128-bit keysize.



**TWINE EXECUTION TIME**

70

60

50

40

30

20

10

0

512

80Bit Key 9.112

128BitKey 9.733

1024

18.243

19.368

2048

36.192

38.581

3072

54.337

57.773

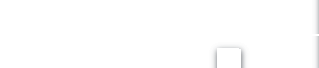
PLAINTEXT SIZE (BYTES)

EXECUTION TIME (MS)

*Figure 7: PICCOLO encryption execution time.*

Finally, TWINE lightweight block cipher is executed and the time taken to encrypt different plaintext sizes with respect to different key sizes are observed and provided in Figure 7 from this we can infer that for both keys, plaintext encryption varies by few milliseconds. This is because both the key sizes use same SBox and feistel function toencrypt.

*Figure 8: 128-bit key execution time comparison of ciphers.*



**128-BIT KEY COMPARISION**

CLEFIA PICCOLO TWINE

400

350

300

250

200

150

100

50

0

512 1024 2048 3072

CLEFIA 12.329 24.782 49.105 73.863

PICCOLO 57.662 115.291 230.445 345.618

TWINE 9.733 19.368 38.581 57.773

PLAINTEXT SIZE (BYTES)

EXECUTION TIME (MS)

Figure 8, provides the comparison results of all three lightweight block ciphers with respect to 128-bit key encryption. We can see that PICCOLO128 is taking more time as the number of rounds is more compared to CLEFIA (Table 2) but even though TWINE is having more number of rounds but the encryption design is just the addition and permutation of plaintext and keysizes.



**80-BIT KEY COMPARISION**

250

200

150

100

50

0

512

PICCOLO 45.32

TWINE 9.112

1024

90.517

18.243

2048 3072

180.865 217.512

36.192 54.337

PLAINTEXT SIZE (BYTES)

EXECUTION TIME (MS)

*Figure 9: 80-bit key execution time comparison of ciphers.*

Figure 9 provides the comparison results of PICCOLO and TWINE with respect to 80-bit key size. TWINE encryption takes minimal time compared to PICCOLO for the same reason mentioned above.

Energy consumption can be calculated in different ways [27]. In this paper, we consider one of the approach where CPUs operating voltage, average current drawn by each cycle in an encryption process is used to measure the energy consumption. For example, in the MCU that we considered, where the CPU is operating at 84MHz frequency, supply voltage of 3.6 volt, and average of 0.0115 Ampere current and if suppose 10000 clock cycles were assumed, then the energy consumed by an operation is 4.93 μA-sec or μ Joule. Following is the mathematical equation used to determine the consumed energy level by an encryption algorithm in thispaper.

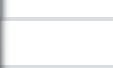
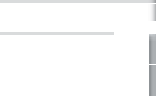
μ Joules / bits

*E = I \* N\* ı \* VCC*

Where I is the average current in ampere, N is the number of clock cycles, ı is the clock period and VCC is the supply voltage [27].

consumes least energy with only 1291.995 μJoules per bit and CLEFIA on the other side burns 1650.71 μJoules per bit when the key size of 128-bit is considered. With 80-bit key size we can see that TWINE performs exceptionally well with only 1214.065 μJoules per bit and PICCOLO performs worst with a higher energy consumption of 6086.42 μJoules perbit.

*Table 3: Number of Clock Cycles for each cipher.*



**ENERGY CONSUMPTION**

9000

8000

7000

6000

5000

4000

3000

2000

1000

0

CLEFIA

80 bit

128bit

192bit

256bit

1650.71

2262.915

2488.5075

PICCOLO

6086.42

7747.72

TWINE

1214.065

1291.995

|  |  |  |
| --- | --- | --- |
| **Algorithms KeySizes ClockCycles** | | |
| CLEFIA | **128-bit** | 3,349,261 |
| **192-bit** | 4,591,406 |
| **256-bit** | 5,049,170 |
| PICCOLO | **80-bit** | 12,349,197 |
| **128-bit** | 15,719,935 |
| TWINE | **80-bit** | 2,463,314 |
| **128-bit** | 2,621,436 |

Cortex M4 provides a Data Watchpoint and Trace (DWT) unit which can be used to trace and setup any benchmarks. By assuming there is a fixed energy consumption for each clock cycle we can get the total number of clock cycles executed for a function from the MCU’s inbuilt register DWT\_CYCCNT by initializing as follows [26]:

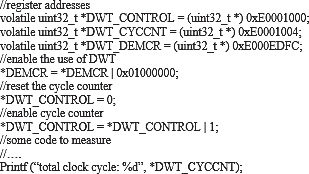


Table 3 provides the average clock cycles required to perform an encryption operation in each of the ciphers. Figure 10 shows the amount of energy consumed by each of the block ciphers from different key sizes in μJoules per bits. Figure clearly shows that PICCOLO takes more energy to compute encryption as the number of clock cycles required to execute is more when compared to other ciphers. ConverselyTWINE

*Figure 10: Energy usage comparison of ciphers.*

Memory compromises of the RAM which is used to execute programs and the flash memory (ROM) which consists of programming and data flash memory, where programming flash memory contains the program code for specific application and data flash memory stores any sensing data or temporary data like the look-up tables if present [15]. Online mbed IDE provides a good GUI based memory representation which is used to compare the memory allocation [17] of the specified block ciphers. Efficient usage of memory is the key role in IoT environments as this is directly related to the operational speed and throughput of the system.



TWINE

25.1

1.3

PICCOLO

25.6

1.5

CLEFIA

27.5

1.3

0

ROM RAM

1.3

1.5

1.3

25

20

15

10

5

25.1

25.6

27.5

30

**MEMORY CONSUMPTION**

KBytes

*Figure 11: Memory usage comparison of ciphers.*

Figure 11 gives the amount of memory needed by CLEFIA, PICCOLO and TWINE. TWINE requires little memory both in RAM and ROM as it has simple round function which does not store more lookup tables to fetch data often. But CLEFIA has less memory efficiency because it contains two different Feistel functions and two different SBox and diffusion matrix which causes more lookups to perform an encryption also since the key sizes used by the lightweight block cipher is more (128, 192, 256 bits) compared to other two block ciphers it is reasonable to accept this high memory consumption as it provides more security (more the key size higher the security) [28]. Memory efficiency of PICCOLO is good compared to CLEFIA but worst when compared with TWINE, this is because even though PICCOLO takes two SBox layers both are just substitution operation and a permutation which is done with one diffusion matrix. So, TWINE is considered as the suitable lightweight block cipher where the memory constraints are high.

Throughput is the metric used to measure amount of data a hardware system can ideally process in a given interval of time. In this paper, we calculate the throughput of each of the lightweight block ciphers to find out which stands best with respect to the resource constraint real world environment where encryption throughput becomes the key metric. To calculate the throughput, number of cycles taken by the encryption process is first calculated later this value is divided with the block size of the algorithm to get total encryption cycles per bit.

Nunber of cycles

Figure 12 reprints the value calculated for each of the lightweight block ciphers. We can see that CLEFIA (128, 192, 256-bit keys) has the highest throughput with an average throughput of 4Kbps. Whereas TWINE (80, 128-bit keys) as a throughput of 4.99Kbps and PICCOLO has 0.6Kbps, this is because the number of clock cycles per bit required to perform an encryption operation in PICCOLO is more compared to other block ciphers and it is indirectly proportional to the throughput of the lightweight blockciphers.

When we compare all three block ciphers with 128-bit encryption then CLEFIA stands first with 5Kbps as the number of clock cycles per bit required to compute an encryption is less, then TWINE stands second with 3.2Kbps and last PICCOLO with a throughput of 0.5Kbps. With 80-bit key encryption we can see that PICCOLO still stands at last with 0.6Kbps when compared to TWINE with 3.4Kbps. Since, PICCOLO has the least throughput we can say that there will be more delay and more energy spent to perform an encryption with real time application in an adhoc environment and therefore this lightweight block cipher is not a suitable candidate to be selected. So, we choose CLEFIA as the best choice.

## CONCLUSION

Internet of things faces numerous challenges like bandwidth, security, privacy, power, scalability and many more among which privacy and security is the most important things to be

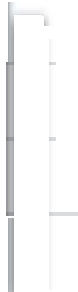
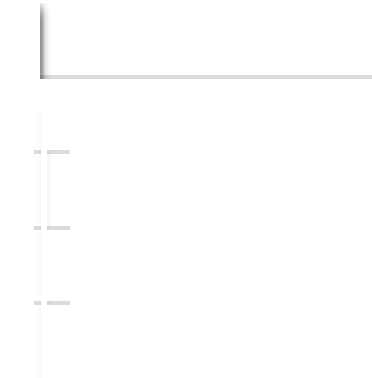
Encryption (cycles/bit) =

Block size

considered in this environment as we cannot trust all the users in IoT. There are numerous crypto block ciphers available but due

Since our MCU runs under 84MHz which means that there can be 84,000,000 cycles getting executed each second. So, the throughput of the encryption function of each lightweight bock ciphers is calculated as follows.

Throughput = CPU Speed Encryption(cycles/bit)



**THROUGHPUT EFFICIENCY**

6

5

4

3

2

1

0

128bit

192bit

256bit

3.316

Kbits / sec

to resource constraints, lightweight cryptographic algorithmsare

chosen to be an ideal candidate for these environments. This papers provides a benchmark performance analysis on STM32F MCU with respect to energy consumption, throughput, execution time and memory consumption which plays an important role in choosing ideal lightweight block ciphers for resource constrained environments. When execution time or the operation time of an encryption function becomes the deciding factor for some applications like In-Vehicle devices, or the industrial control systems in an IoT environment then TWINE is the suitable candidate with key size of 80-bit, however when we consider 128-bit for more security [28] then CLEFIA turns out to be the suitable ciphers for small sized plaintext and as the size increases TWINE becomes the ideal candidate for encryption. Applications like RFID, sensor nodes, medical/healthcare devices etc. requires less memory consumption and less energy consumption so that the operational speed increases, TWINE is a best solution followed by CLEFIA which differs only by few metric units. In IoT environment, throughput matters a lot because higher the throughput lesser the delay andtherefore

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CLEFIA | PICCOLO | TWINE | hardware can spend less energy and later go into sleep mode |
| 80 bit |  | 0.677 | 3.398 | which prolongs the battery power. CLEFIA is the best choice for |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | 3.193 encryption where higher throughputmatters. |
|  |  |  | 4.999 | 0.532 |  |
|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  | 3.646 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

*Figure 12: Throughput comparison of ciphers.*

## REFERENCES

1. Meola, Andrew. "How the Internet of Things will affect security & privacy." *Business Insider*. Business Insider, 19 Dec. 2016. Web. 28 Feb.2017.
2. Stallings, William. *Cryptography and network security: principles and practice*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006.Print.
3. "Lightweight Cryptography." *CryptoLUX> Lightweight Cryptography*. N.p., n.d. Web. 28 Feb.2017.
4. Suzaki, T., Minematsu, K., Morioka, S., & Kobayashi, E. (2011). *Twine: A lightweight, versatile block cipher.* In ECRYPT Workshop on Lightweight Cryptography (pp. 146169).
5. Wang, Y., Wu, W., & Yu, X. (2012). *Biclique cryptanalysis of reduced-round piccolo block cipher*. In Information Security Practice and Experience (pp. 337-352). Springer Berlin Heidelberg. pdf atspringer
6. Minier, M. (2013). *On the Security of Piccolo Lightweight Block Cipher against Related-Key Impossible Differentials*. In Progress in Cryptology–INDOCRYPT 2013 (pp.308-318).

Springer International Publishing.pdf at springer.com

1. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., &Shirai, T. (2011). *Piccolo: an ultra-lightweight blockcipher*. In Cryptographic Hardware and Embedded Systems–CHES 2011 (pp. 342-357). Springer Berlin Heidelberg. pdf atspringer
2. Çoban, M., Karakoç, F., &Boztaş, Ö. (2012). Biclique cryptanalysis of TWINE. In Cryptology and Network Security (pp. 43-55). Springer Berlin Heidelberg. pdf ateprint.iacr.org
3. Wang, Y., & Wu, W. (2014, January). *Improved Multidimensional Zero-Correlation Linear Cryptanalysis and Applications to LBlock and TWINE*. In Information Security and Privacy (pp. 1-16). Springer International Publishing. pdf atspringer.com
4. Website of the International Standard Organization,<http://www.iso.org/iso/iso_catalogue/catalogue>

\_tc/catalogue\_detail.htm?csnumber=56552

1. Li, Y., Wu, W., & Zhang, L. (2012). *Improved integral attacks on reduced-round CLEFIA block cipher*. In Information Security Applications (pp. 28-39). Springer Berlin Heidelberg. pdf atspringer
2. Tezcan, C. (2010). *The improbable differential attack: Cryptanalysis of reduced round CLEFIA*. In Progress in Cryptology-INDOCRYPT 2010 (pp. 197-209). Springer Berlin Heidelberg.
3. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., & Iwata, T. (2007, January). *The 128-bit blockcipher CLEFIA*. In Fast software encryption (pp. 181-195). Springer Berlin Heidelberg.
4. EmbeddedC."[https://w](http://www.engineersgarage.com/tutorials/eme)ww.engin[eersgarage.com/tutorials/eme](http://www.engineersgarage.com/tutorials/eme)bedded-c-language”
5. "Memory Model - Handbook | mbed." *Memory Model - Handbook | mbed*. N.p., n.d.Web.
6. "Benefits of C/C over Other Programming Languages." *Invensis Blog*. N.p., n.d.Web.
7. "Memory Model - Handbook | mbed." *Memory Model - Handbook | mbed*. N.p., n.d.Web.
8. "Mbed Compiler - Handbook | mbed." *Mbed Compiler - Handbook | mbed*. N.p., n.d.Web.
9. "The mbed interface." *The mbed interface - mbed OS 5 Handbook*. N.p., n.d.Web.
10. Suzaki, T., &Minematsu, K. (2010, January). *Improving the generalizedFeistel*.InFastSoftwareEncryption(pp.19-39).

Springer Berlin Heidelberg.pdf at eprint.iacr.org

1. "SystemyRTiembedded."“<http://www.ue.pwr.wroc.pl/system>y\_rt/RTE6.pdf”N.p., n.d.Web.
2. "ARM Cortex-M." *Wikipedia*. Wikimedia Foundation, n.d. Web.
3. "Cortex-M4 Processor." *Cortex-M4 Processor - ARM*.N.p.,

n.d. Web.

1. “<http://www.st.com/en/microcontrollers/stm32-32-bit-arm->cortex-mcus.html?querycriteria=productId=SC1169” N.p.,

n.d. Web.

1. "STM32F401RE."*STM32F401RE - STM32 Dynamic Efficiency MCU, ARM Cortex-M4 core with DSP and FPU, up to 512 Kbytes Flash, 84 MHz CPU, Art Accelerator - STMicroelectronics*.N.p.,n.d.Web. [http://www.st.com/en/microcontrollers/stm32f401re.html.](http://www.st.com/en/microcontrollers/stm32f401re.html)
2. "ARM Information Center." *ARM Information Center*. N.p., n.d.Web. [http://infocenter.arm.com/help/index.jsp?topic=%2Fcom.arm.](http://infocenter.arm.com/help/index.jsp?topic=%2Fcom.arm) doc.ddi0337h%2FBIIFBHIF.html.
3. "Studying the Effects of Most Common Encryption Algorithms."(n.d.):n.pag.Web. <http://www.iajet.org/iajet/iajet_files/vol.2/no.1/Studying%20t> he%20Effects%20of%20Most%20Common%20Encryption% 20Algorithms.pdf.
4. "Key size." *Wikipedia*. Wikimedia Foundation, n.d.Web.
5. Kmarquet. "Kmarquet/bloc." GitHub. N.p., n.d. Web. https://github.com/kmarquet/bloc/tree/master.
6. "Sony Corporation Global Headquarters." SONY. N.p., n.d.We[b.http://w](http://www.sony.net/Products/cryptography/clefia/downl)ww[.sony.net/Products/cryptography/clefia/downl](http://www.sony.net/Products/cryptography/clefia/downl)oad/index.html.