RESEARCH REVIEW JOURNALS
Volume-04
Issue-03
March-2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com[UGC Listed Journal]

Performance Analysis of Light Weight Security Algorithms for Resource Constrained Devices

Khan Riaz

Assistant Prof., Computer Science and Engineering, Islamic University of Science and Technology, Kashmir (India)

ARTICLE DETAILS

Article History
Published Online: 13 March2019

Keywords
IoT, Light weight security, Constrained devices, sensor networks, Raspberry Pi.

***Corresponding Author**
Email:riazk3[at]gmail.com

ABSTRACT

With the advent of technologies like Internet of Things (IoT) and Machine to Machine (M2M) communication, a huge quantity of data is generated every day. Being a distributed system of constrained devices, this data needs to be communicated securely without wasting the resources of constrained devices. Therefore there is need of appropriate lightweight security protocols to avoid the security threat to future internet. In this context, a study of lightweight security algorithms is presented in this paper. The algorithms are first theoretically analyzed followed by their implementation on Cryptool and Raspberry Pi in order to check their efficacy.

1. Introduction

Internet of Things (IoT) is a novel worldview that is quickly making progress in the field of cutting-edge remote media communication [1]. IoT is a global movement that unites people, data, processes and things to build network connections that are more pertinent and useful than ever before. It is a structure of interconnected computing items, such as RFID tags, sensors, actuators, and cell phones; digital machines; and people that offer the facility of transferring data among networks without need of human-to-computer or human-to-human interactions.

As IoT is growing rapidly, it faces risks and challenges, such as how to handle huge amounts of data, processing power, deal with energy consumption, address security threats, and how to encrypt/decrypt huge data [1].

IoT helps in creating connections between dissimilar things present in heterogeneous environment. This kind of openness and very less human intervention can make IoT exposed to number of attacks like man in middle attack, Denial of Service (DoS) attack. Moreover, any device can have access the network that leads to unauthorized access. These attacks can damage device physically and network connections too. This will ultimately compromise the security and privacy of IoT.

To address these challenges when many smart devices are connected in an IoT environment, there is an increasing demand for the use of appropriate cryptographic solution into the embedded applications. However, these smart devices usually have limited resources with low computational power, low battery life, smaller size, limited memory and power supply. Hence, the conventional cryptographic primitives might not be suited for low-resource smart devices. Therefore in such applications lightweight cryptography is introduced that provides solutions suitable for constrained devices [2].

Furthermore, IoT has also exposed many security attacks that can damage the network connection due to an unauthorized access. This leads to the security parameters and network privacy being compromised. In addition, IoT utilizes the cloud computing concept, which has many security issues and challenges [2, 3]. Apart from these issues, the resource-constrained devices, which have less computational power, limited battery life, a small amount of memory, and low bandwidth, need an efficient security solution that will not crunch the resources of IoT.

Therefore this paper presents a study of the lightweight algorithms suitable for resource-constrained devices that form the bulk of the IoT setup. Algorithms including AES, Mickey 2, Grain, Rabbit and TEA were first simulated using an Open-Source Software tool viz., Cryptool and later implemented using Java programs on Raspberry pi.

The remainder of this paper is organized as follows: Section 2 presents the background and survey of the field, section 3 details the work carried out, section 4 gives the experimental results and comparison both using Cryptool and the Java platform. Finally the paper is concluded in section 5.

2. Background

In [4, 5], Cryptography is defined as an ancient art of writing secret with the knowledge of science. Cryptography was first used in writing long dates back to circa 1900 B.C. where it was used as non-standard hieroglyphs in an inscription by an Egyptian. Some specialists claim that cryptography came into existence suddenly after writing was developed, with kind of applications such as political letters to war-time battle tactics. With the development of computer communication the new forms of cryptography was discovered as surprise to many intended users. When communicating over an insecure and untrusted medium such as internet, cryptography is necessary to secure the data under transmission.

There are five primary functions of cryptography today:

i) Privacy/confidentiality: This function ensures that only intended receiver can read the message and no one else.
ii) Authentication: It helps in providing one's identity.

iii) Integrity: This function guarantees the receiver that the received data has not been altered when under transmission over the untrusted medium.

iv) Non-repudiation: This function facilitates that the receiver cannot deny of reception of data and also provides a mechanism to verify that the sender has sent a message.

v) Key exchange: It is the mechanism of sharing the secret keys between sender and receiver.

In cryptography, the encryption and decryption is based upon the type of cryptographic scheme being employed and some form of key. The process may be depicted as under:

$$C = E(k, P) \quad (1)$$

$$P = D(k, C) \quad (2)$$

Where, P = plaintext, C = cipher-text, E = the encryption method, D = the decryption method, and k = the key.

2.1. Cryptographic Algorithms

There are different types of cryptographic algorithms that are being used to secure a system. In this paper, they are categorized on the basis of keys that are used for encryption and decryption, type of application and its use. Following are the types of algorithms that are commonly used (see fig. 1):

i) Secret Key Cryptography (SKC): Another name of symmetric key encryption that uses a single key for both encryption and decryption to provide confidentiality and secrecy to a system.

ii) Public Key Cryptography (PKC): It is also known as asymmetric cryptography that uses one key for encryption and another different key for decryption to provide authentication, non-repudiation, and key exchange mechanism for symmetric encryption.

iii) Hash Functions: It provides a digital fingerprint by performing mathematical alterations to irrevocably "encrypt" the message and hence provides integrity.

In this paper we consider SKC and PKC only and Hash Functions are ignored due to their limited use in present applications of smart devices.

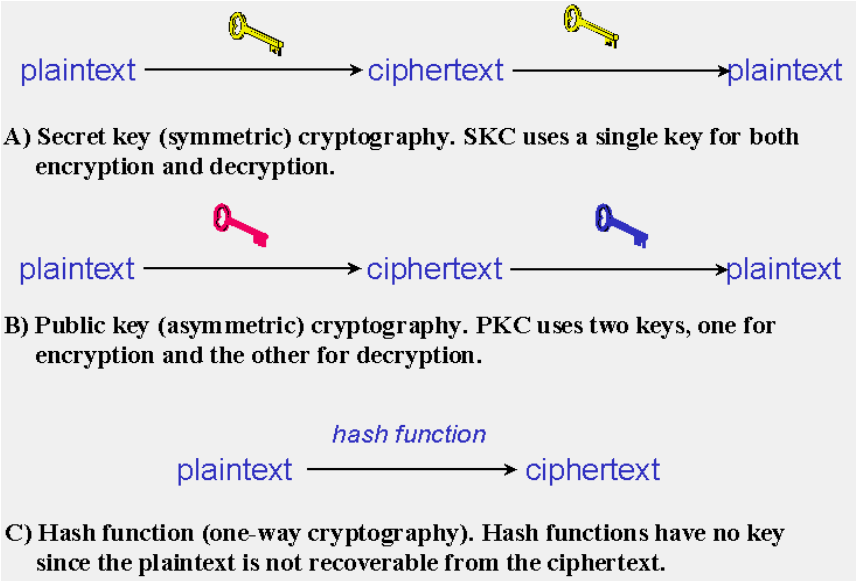

Fig. 1 Pictorial Representation of Cryptographic Algorithms

2.1.1. Secret Key Cryptography

As stated above that SKC is also known as symmetric cryptography which employs a single key for encrypting and decrypting a message.

In this type of cryptography, it is evident that the secret key has to be well-known to both the sender and the receiver before they communicate. The key distribution in SKC is the biggest difficulty for employing such cryptographic approach.

SKC schemes are usually characterized as either block ciphers or stream ciphers as shown in fig. 1 and fig. 2 respectively.

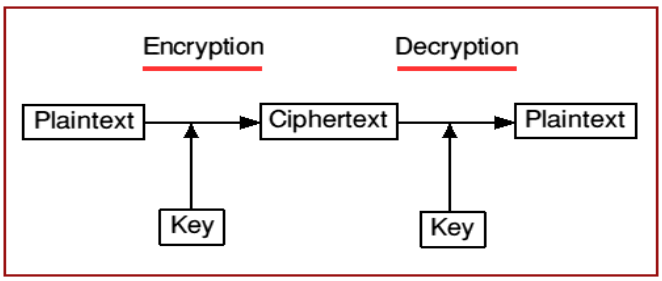

Fig. 2. Use of SKC in Block Cipher

A stream cipher is designed to work on a single bit (byte or computer word) at a time and uses some sort of mechanism providing feedback that helps in changing the key constantly.

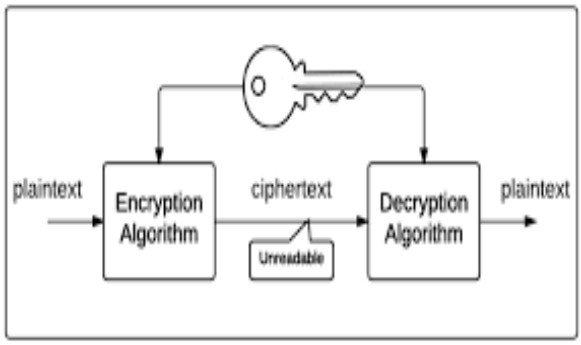

Fig. 3. Stream Cipher (Encryption and Decryption)

In block cipher approach one block of data is encrypted at a time by using the same single key on each block. In general, the encryption of same plaintext by using same key each time would yield the same cipher-text in block cipher whereas it is not the case in stream cipher where same plaintext would produce a different cipher-text. Fiestel cipher is the most common type for block encryption algorithms, named after cryptographer Horst Feistel. As shown in Fig. 4, a Feistel cipher is obtained by combining substitution, permutation (transposition), and key expansion techniques which generate lot of "confusion and diffusion" in the plaintext. Fiestel design provides an advantage by having similar encryption and decryption phases, that just require only a reverse of the key operation applied, thus affectedly reduces the size of the code (software) or circuitry (hardware) needed to execute the cipher.

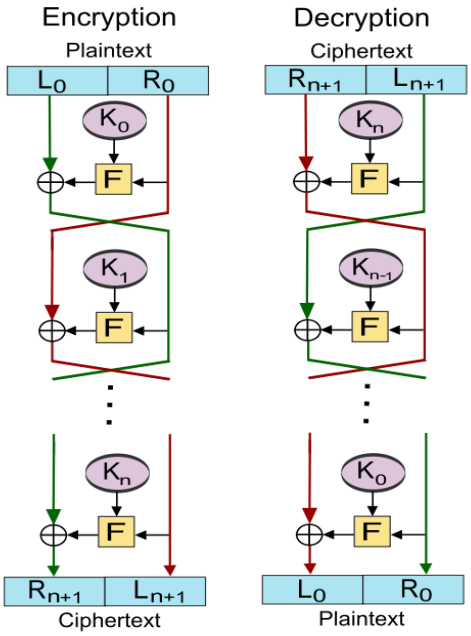

Fig. 4: Feistel cipher

Secret key cryptography algorithms in use today or, at least, important today even if not in use include:

- Data Encryption Standard (DES): DES was designed in the 1970s by IBM and accepted by the National Bureau of Standards (NBS) for commercial and uncategorized government applications in 1977. It is considered as one of the most suitable and appropriate SKC schemes. DES uses a 56 bit key with feistel structure to operate on a 64-bit block. Due to the complex collection of rules and alterations in DES, it presents a design that explicitly yields fast hardware executions with slow software executions, while as it is not substantial in present time since the speed of today's computer processors is many times faster than the ones twenty years ago. It is also believed that DES was based on an earlier cipher called Lucifer which as per reports had a 112-bit key. But it was rejected partly so as to fit the algorithm onto a single chip and also because of the National Security Agency (NSA). The NSA further proposed many new twists to DES that were thought by experts to be introduced in order to weaken the cipher, but later on in the 1990s an analysis showed that the NSA proposals in fact toughen the DES [5].
- Advanced Encryption Standard (AES): It was initiated by NIST in 1997 in public domain in order to develop a new and secure cryptosystem for U.S. government. The AES became successful as the certified successor to DES in the year 2001 that uses a single key cryptographic scheme called Rijndael, a kind of block cipher proposed by Joan Daemen and Vincent Rijmen. The algorithm has flexibility of using a variable length block as well as key. The latest description certified that any combination of keys lengths of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits can be used. Thus FIPS PUB 197 pronounces a 128-bit block cipher engaging a 128-, 192-, or 256-bit key.

2.1.2. Public Key Cryptography

Public key cryptography (PKC) has been believed to be the greatest and substantial growth in cryptography in the last 300-400 years. A Stanford University professor Martin Hellman and graduate student Whitfield Diffie in the year 1976 defined modern PKC publicly. Their paper presented that two parties can have secure communication over a non-secure and untrusted communication channel without the need of sharing a secret key.

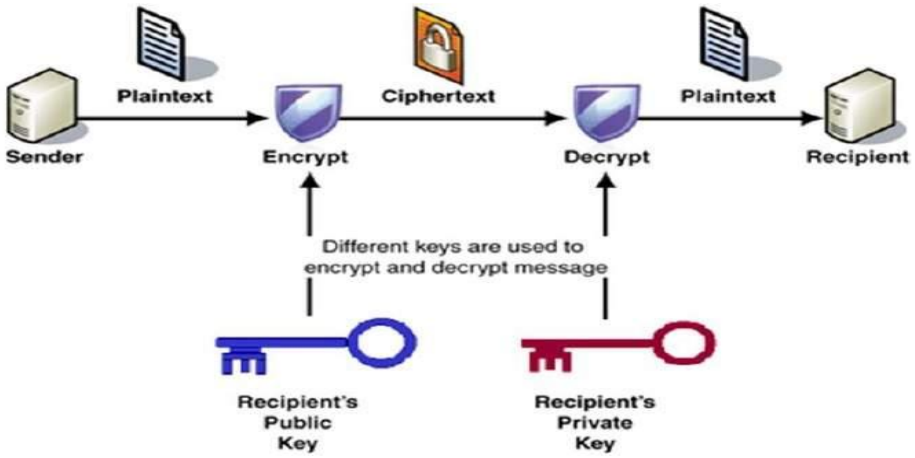

Fig. 5. Public Key Cryptography

The concept of PKC depends upon one-way functions. These are mathematical functions that are easily computed but to find their inverse is comparatively problematic to calculate. The two keys in PKC are mathematically interrelated while having knowledge about one key will not let someone to easily deduce the other key. In this scheme, one key is used to encrypt the plaintext whereas the other one is used to decrypt the cipher-text. The important thing is that it is not mandatory to use a particular key for encryption or decryption as both keys are used in pair and the approach is also called asymmetric cryptography as shown in fig. 5.

In PKC, one of the keys is nominated as public key and it can be publicized extensively. The other key is elected as the private key and is never exposed to any other party in the network. It is a traditional scheme to forward messages between Alice and Bob. Suppose Alice desires to send Bob a message, Alice will encrypt its message using Bob's public key; Bob decrypts the cipher-text using his private key. This method can also be used to prove the authenticity of message and avoids non-repudiation as well. For example, Alice could encrypt some plaintext with her private key; when Bob decrypts using Alice's public key, he knows that Alice sent the message (authentication) and Alice cannot deny having sent the message (non-repudiation).

PKC algorithms that are still used for key exchange or for creating digital signatures include RSA (Rivest Shimmer and Adleman), Diffie-Hellman, Digital Signature Algorithm (DSA), ElGamal and Elliptic Curve Cryptography (ECC).

2.2. Lightweight Cryptography

Lightweight cryptography (LWC) aims to target a wide range of devices that can be implemented on an extensive band of hardware and software. On the high end of the device band there are servers and desktop hosts followed by small tablets and smartphones. Conventional security algorithms usually perform well in such devices whereas on the lower spectrum end the devices such as sensors, RFID tags and embedded systems where LWC algorithms can be useful [6].

For RFID tags that are considered as constrained devices, not battery-operated, with very limited power supply require security algorithms that will not only use a very small amount of electronics, but also meet severe timing and power necessities [7, 8].

The prime objective of LWC is to target the devices at the lower spectrum end but it should be able to execute lightweight security algorithms at the high end of the band as well. For example, many resource-constrained devices such as sensors can send information to an aggregator which may or may not be constrained. But, the aggregator must have a support for lightweight security algorithms to provide interoperability with the constrained devices when they use LWC algorithms [5].

In cryptography, one of the state-of-the-art schemes is LWC that is emerged for execution in constrained environments including sensors, RFID tags, contactless smart cards, health-care devices etc. furthermore, LWC also offers sufficient security and is not always susceptible to feat the security-efficiency trade-offs.

Additionally, lightweight cryptography also has its share of research in both Symmetric and Asymmetric Cryptography

2.2.1. Lightweight Symmetric Key Cryptography

Block Ciphers: Many block ciphers with lightweight characteristics have been proposed since the AES developed. PRESENT [9] and TEA [10] are well-studied and popular algorithms among various block ciphers due to their security and implementation. These ciphers are all set to use in real-world systems and have been considered in this paper for study.

PRESENT is a lightweight block cipher, developed by the Orange Labs (France), Ruhr University Bochum (Germany) and the Technical University of Denmark in 2007.

The algorithm has distinguished characteristics due to its compact size and is designed by Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe [9].

It is a sort of variable key length lightweight block cipher algorithm that takes 64-bit block with 80-bit and 128-bit key length support. The algorithm uses the SPN structure, whose operation is mainly divided into two parts, key extension operation and a total of 31 rounds of iteration operation. Each

iteration function F is composite of three different transformations: addRoundKey, S-box and PLayer. KeySchedule is used to produce the round key which is used for round reiteration process.

The 64-bit plaintext (P) after 31 rounds of iteration operation and the last round XOR with the round key, is used to get 64-bit ciphertext C. The different transformations operate on the intermediate result, called the State.

Tiny Encryption Algorithm (TEA) is remarkable for its easiness of explanation and implementation, typically consist of a very few lines of code. It was designed in Cambridge Computer Laboratory by David Wheeler and Roger Needham. For the very first time, it was offered at the "Fast Software Encryption workshop in Leuven in 1994".

TEA works on two 32-bit unsigned integers that can be derivative of a 64-bit data block and it supports a 128-bit key. It employs a Feistel type structure with a recommended 64 rounds, normally implemented in pairs labeled cycles. It has an extremely simple key schedule, mixing all of the key material in exactly the same way for each cycle. In order to avoid simple attacks based on the round symmetry different multiples of magic constants are used.

Stream Ciphers

MICKEY 2 [11], Grain V1 [12] and Rabbit [13] are the widely used choices and have been considered for simulation. Each of these are briefly described below:

Mutual Irregular Clocking KEYstream generator (MICKEY): is designed for hardware platforms and exhibit stream cipher characteristics with limited resources. The algorithm is not patented and is free for any use.

MICKEY 2 cipher maps a key of 80-bit and an initialization vector (IV) with variable length (0 to 80 bits) to a keystream which has a maximum of 240 bits length. The key-stream generator utilizes the two registers R and S (100 bits each) [11].

Grain - A Stream Cipher for Constrained Environments. It accepts a key of 80-bit and an IV of 64-bit. The stipulations do not suggested a maximum length of output per (key, IV) pair. A number of possible weaknesses have been recognized in the cipher and fixed in Grain 128a which is now the suggested cipher that can be used for hardware environments to provide both 128 bit authentication and security.

Grain's internal state with 160-bit contains an 80-bit linear feedback shift register (LFSR) and a non-linear feedback with 80-bit shift register (NLFSR). Grain updates one bit of both LFSR and NLFSR state for every cipher-text bit released by a nonlinear filter function. The 80-bit NLFSR is updated with a nonlinear 5-to-1 Boolean function and a 1 bit linear input selected from the LFSR. The nonlinear 5-to-1 function receipts as input of 5 bits of the NLFSR state. The LFSR with 80 bits is updated with a 6-to-1 linear function. During keying process the output of the cipher is additionally fed-back into both the NLFSR and LFSR update functions as linear inputs. In the original Grain Version 0.0, four bits of the 80-bit LFSR and one bit of the 80-bit NLFSR are provided to a nonlinear 5-to-1 Boolean function. The output is combined with 1 bit of the 80-bit NLFSR linearly and released out as output.

Four bits of the 80-bit LFSR and one bit of the 80-bit NLFSR and are provided to a 5-to-1 non-linear Boolean function and the output is combined with 7 bits of the 80-bit NLFSR linearly and released out as output in the updated Grain Version 1.0 submission of Grain. To set the cipher, the 80-bit key is loaded directly into the 80-bits NLFSR and the 64-bit IV is loaded into the low 64-bits of the LFSR and the remaining 16 high bits of the LFSR are filled with ones. The cipher is wrapped for 160 rounds where the 160 key-stream bits produced are fed-back linearly into both the NLFSR and LFSR update functions. The cipher releases no key-stream output during the initialization process.

Rabbit was first presented at the "Fast Software Encryption workshop in 2003" and is a synchronous stream cipher [13]. In Rabbit until now, no cryptographical weaknesses have been exposed. The algorithm takes an IV of 64 bit and 128 bit secret key as input and produces an output block of 128 pseudo-random bits for each iteration from a mixture of the internal state bits. Encryption as well as decryption is done by XOR'ing the pseudo-random data with the plaintext/cipher-text. The size of the internal state is 513 bits divided between eight 32-bit state variables, eight 32-bit counters and one counter carry bit. The eight coupled non-linear functions are used to update eight state variables. The counters confirm a lower bound on the period length for the state variables. Rabbit was designed to justify a key size of 128 bits for encrypting up to 264 blocks of plaintext and is considered faster than commonly used ciphers. This suggests that for an attacker who doesn't have knowledge of the key, should not be able to distinguish up to 264 blocks of cipher output from the output of a truly random generator, using fewer steps than would be needed for an thorough key search over 2128 keys.

2.2.2. Lightweight Asymmetric Key Cryptography

While in smart object networks, lightweight public key primitives are in great demand for key management protocols. The compulsory resource for public key primitives is too larger than that of symmetric key primitives. At this time, there are no promising primitives that meet adequate security and lightweight characteristics equated with the conservative primitives such as RSA [14] and Elliptical Curve Cryptography (ECC) [15]. Certain public key primitives (e.g. ECC) can be realized with relatively small footprint, but they cannot implement within a practical time limit.

Elliptic Curve Cryptography (ECC) was revealed by V. Miller (IBM) and N. Koblitz (University of Washington) in the year 1985 as an alternate tool for employing public-key cryptography (see fig. 6).

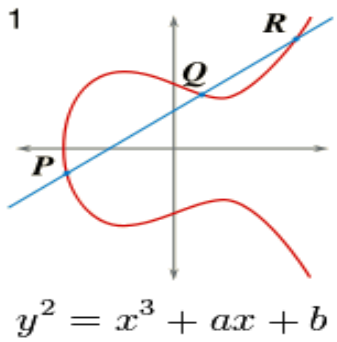

Fig 6. Elliptical Curve Cryptography

The elliptic curve equation is presented as:

Following terms will be used in the equation are::

E -> An Elliptic Curve

P -> A Point on the curve

n -> The Maximum limit (This needs to be a prime number)

Key Generation

In key generation, we have to generate both public key and private key which is an important part of ECC. The sender uses receiver's public key to encrypt the message and the receiver will decrypt the message with its private key.

Now, the next step is to select a number 'd' within the range of 'n'.

We can generate the public key by using the following equation

Q = d * P

d = It defines the random number that we have selected within the range of (1 to n-1).

P is the point on the curve.

'Q' is the public key and 'd' is the private key.

Encryption:

Let us consider 'm' as message to be sent. This message needs to be represented on the curve. This has implementation details in-depth. The company named certicom does all the advanced research on ECC.

Suppose 'm' has the point 'M' on the ECC curve 'E'. Next select randomly letter 'k' from [1 – (n-1)]. There will be generated two cipher texts and let these be C1 and C2.

C1 = k * P

C2 = M + k * Q

C1 and C2 are the cipher-text to be sent.

Decryption:

We need to deduce back the message 'm' that was initially sent to us,

M = C2 – d * C1

M is the original message that we had sent.

Proof

M = C2 – d * C1

M can be represented as 'C2 – d * C1'

C2 – d * C1 = (M + k * Q) – d * (k * P) (C2 = M + k * Q and C1 = k * P) = M + k * d * P – d * k *P (canceling out k * d * P)

= M (Original Message sent)

3. Work Carried Out

In this paper, the work was carried out is in three phases viz. theoretical study, simulation study and implementation.

3.1. Theoretical Study

The Light weight algorithms viz. PRESENT, TEA, MICKEY, Grain and Rabbit were studied and analyzed against AES. Some of the main findings are presented here.

PRESENT:

Key sizes	80 or 128 bits
Block sizes	64 bits
Structure	SPN
Rounds	31

TEA

Key sizes	128 bits
Block sizes	64 bits
Structure	FS
Rounds	32 used in simulation though it is variable

AES

Key sizes	128/192/256 bits
Block sizes	128 bits
Structure	SPN
Rounds	10/12/14 depending on key size

Mickey

Key sizes	80 bits
IV	0-80 bits
Stream	max of 240
Rounds	NA

Grain

Key sizes	80 bits
IV	64 bits
Stream	128 bits
Rounds	16

Rabbit

Key sizes	128 bits
IV	64 bits
Stream	128 bits per iteration Rounds

3.2. Simulation Study

For the simulation of above cite algorithms we used CrypTool [16] which is an open source project and expected to implement more than 400 algorithms. These include all-most all classical ciphers, modern symmetric and asymmetric ciphers which include ECC, hybrid encryption, RSA, digital signatures, homomorphic encryption and Diffie–Hellman key exchange.

In this paper, six lightweight algorithms are simulated. The results are analyzed in next section.

3.3. Implementation

LWC Algorithms were implemented using Java programming and run on Raspberry Pi for evaluation of latency and analysis for future hardware implementation for use in varying applications.

The Raspberry Pi [17] is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation.

"Several generations of Raspberry Pi's have been released. All models present a Broadcom system on a chip (SoC) along with an integrated ARM friendly central processing unit (CPU). There is on-chip graphics processing unit (GPU) as well. Processor speed varies from 700 MHz to 1.4 GHz for the Pi 3 Model B+; on-board memory ranges from 256 MB to 1 GB RAM. Secure Digital (SD) cards store the operating system and program memory in either SDHC or MicroSDHC. The boards also have one to four USB ports. For output of video, HDMI and composite video are sustained, with a standard 3.5 mm phono jack for audio output. The B-models have an Ethernet port and the Pi 3 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth".

Our implementation consists of following steps:

i) Integration of Raspberry Pi with Laptop through VNC Server.
ii) Java installation on Raspberry Pi
iii) Writing code for implementation of cryptographic algorithms in text editor available
iv) Execution of the algorithms
v) Data collection for comparative analysis

4. Experimental Results and Comparison

The simulation for the selected cryptographic algorithms was performed using Cryptool. Each algorithm was run 10 times and an average was calculated for each algorithm for the total time including encryption and decryption.

The behavior of the algorithms is depicted in the table 1 and graph in fig.7. It is observed that the maximum time is utilized by Grain v1 and minimum time is utilized by MICKEY v2. The algorithms like AES and TEA showed average behavior.

Table – 1
Execution Time Comparison using Cryptoolv2.1

Algorithm	Execution Time in ms
GRAINv1	1047
AES	1020
TEA	1038
MICKEYv2	1018
PRESENT	1025
RABBIT	1024

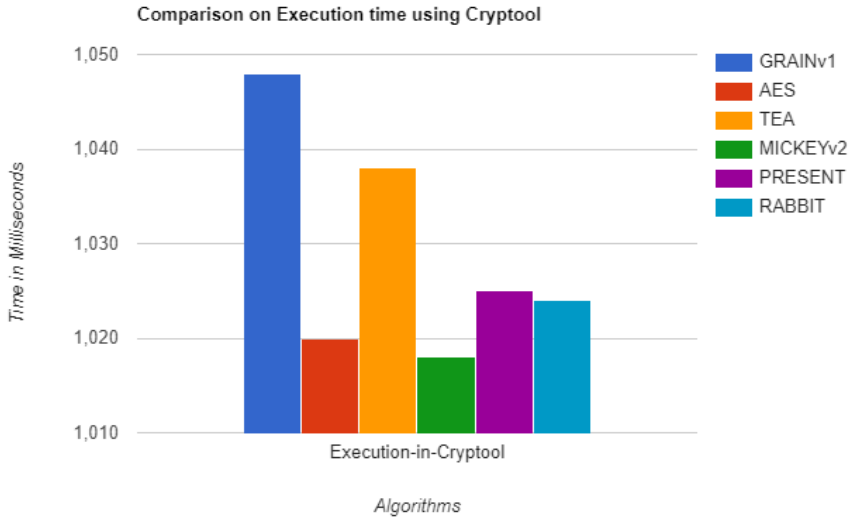

Fig. 7. Comparison of Execution time (Encryption and Decryption)

For implementation on Raspberry Pi, four algorithms were used. These included 3 already tested using Cryptool and ECC was included due to its wide usage in smart application nowadays. The programs for the selected cryptographic algorithms were run on Raspberry Pi. Each algorithm was run 10 times and an average was calculated for each including encryption and decryption.

The behavior of the algorithms is depicted in the table 2 and graph in fig. 8. It is observed that the maximum time is utilized by Grain v1 and minimum time is utilized by ECC. The algorithms like AES and TEA showed average behavior.

Table- 2:
Execution Time Comparison using Raspberry Pi

Algorithms	Encryption Time(ms)	Decryption Time(ms)	Total Time (ms)
GRAIN v1	3688	14771	18459
EEC	17	21	38
TEA	71	45	116
AES	1089	2	1091

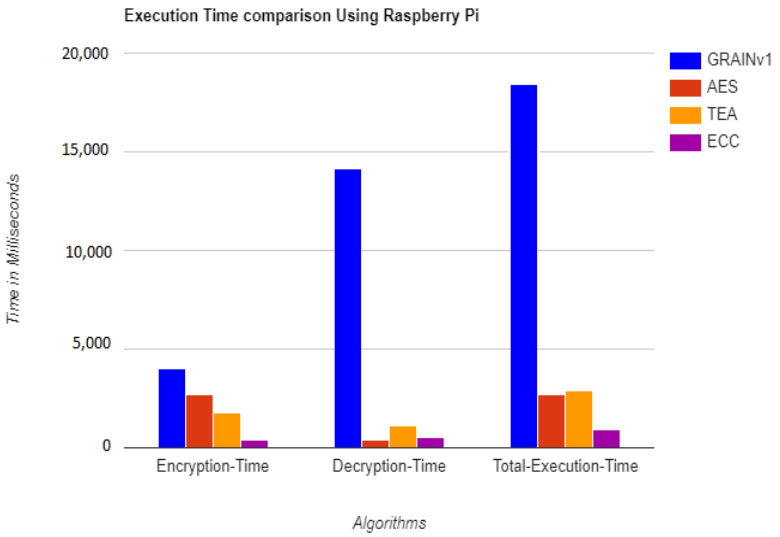

Fig. 8. Comparison of Execution time using Raspberry Pi

5. Conclusion

The study in this paper looked at Lightweight Cryptographic algorithms that are best choices for resource constrained devices being used in IoT platforms. The results of simulation and implementation depict that algorithms with smaller key sizes are better than ones with larger key sizes. The algorithms are good choices for both software and hardware implementation in smaller devices. MICKEYv2 is an ultra-lightweight algorithm with one of the most compact encryption methods. Due to these characteristics, it is useful in applications of low power consumption and response time. Simulation and Implementation results confirmed better performance by MICKEYv2 and ECC algorithms. Grainv1 algorithm showed maximum execution time and is thereby not suitable for resource constrained environments and applications with requirement of better response time.

References

1. Saurabh Singh, et al, “Advanced Lightweight Encryption algorithms for IoT devices: survey, challenges and solutions”, in International Journal of Ambient Intelligence and Humanized Computing, April 2017.
2. Lu Zhou, et al, “Towards practical white-box light-weight block cipher implementation for IoTs”, in Elsevier Journal of Future Generation Computer Systems, 2018. (article in press).
3. S Zhou, Z Xie, On Cryptographic Approaches to Internet-Of-Things Security, ZTE Corporation, 2016.
4. Bruce Scheiner, “Applied Cryptography: Protocols, Algorithms and Source Code in C”, John Wiley & Sons, 2007.
5. W Stallings, “Network Essentials and Security”, PHI, 5e, 2007.
6. Workshop on Lightweight Cryptography, Oct 2016. https://www.nist.gov/news events/events/2016/10/lightweight-cryptography-workshop-2016.
7. Arbia Riahi Sfar, et al, “A roadmap for Security Challenges in Internet of Things”, Elsevier Journal of Digital Communications and Networks, April 2017.
8. Sooyeon Shin, et al, “ Experimental Performance Analysis of Lightweight Block Ciphers and Message Authentication Codes for Wireless Sensor Networks”, in International Journal of Distributed Sensor Networks, Vol 13(11), Sep 2017.
9. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, “PRESENT: An Ultra-Lightweight Block Cipher.” in CHES 2007, no. 4727 in LNCS, pp. 450–466, Springer-Verlag, 2007.
10. Wheeler DJ, Needham RM, “TEA, a tiny encryption algorithm.”, in Proceedings of International Workshop on Fast Software Encryption, Springer, Berlin, pp 363-366, 1994.
11. Steve Babbage, Mathew Dodd, “The Stream Cipher MICKEY 2.0, ECRYPT Stream Cipher (2006), http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf
12. M. Hell, T. Johansson, and W. Meier, “Grain - a stream cipher for constrained environments.” International Journal of Wireless and Mobile Computing, Special Issue on Security of Computer Network and Mobile Systems., vol. 2, no. 1, pp. 86–93, 2006. NIST.
13. M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius. "Rabbit: A new high-performance stream cipher”, In T. Johansson, editor, Proc. Fast Software Encryption 2003, volume 2887 of LNCS, pages 307–329. Springer, 2003.
14. Zhou X, Tang X, “ Research and Implementation of RSA algorithm for encryption and decryption”, in proceedings of 6th International Forum for Strategic Technology (IFOST): 2011, pp 1118-1121.
15. Sonali U. Nimbhorkar, and Dr L. G. Malik. "A Survey On Elliptic Curve Cryptography (ECC)," International Journal of Advanced Studies in Computers, Science and Engineering Vol. 1, 2012, issue 1 pp. 1-5.
16. Adamovic, Sasa; Sarac, Marko; Veinovic, Mladen; Milosavljevic, Milan; Jevremovic, Aleksandar , "An Interactive and Collaborative Approach to Teaching Cryptology", Educational Technology & Society. 17 (1): 197–205, 2014.
17. "Introducing Raspberry Pi Model B+". Raspberry Pi Foundation. 2014.

