A Light Weight Cryptography (LWC) for Small Scale Data in IoT Devices

Jahidul Arafat
Assistant Professor, Department of Computer Science & Engineering (CSE),
Bangladesh Army International University of Science and Technology (BAIUST)
Cumillla Cantonment-3501,
Cumilla, Bangladesh

Md. Abdul Malek Chowdury
Department of Computer Science and Engineering (CSE),
Bangladesh Army International University of Science and Technology (BAIUST)
Cumillla Cantonment-3501,
Cumilla, Bangladesh

Tazkia Binty Faruque
Department of Computer Science and Engineering (CSE),
Bangladesh Army International University of Science and Technology (BAIUST)
Cumillla Cantonment-3501,
Cumilla, Bangladesh

ABSTRACT

In the digital world, which is currently evolving and changing at such a rapid pace, the security of information has become increasingly more important. To preserve the secrecy of the information, cryptography has specific roles to protect files from unauthorized access. In this paper, a new robust and lightweight cryptography algorithm named as LWC to increase security at a cheaper cost in the Symmetric-key producing algorithm. This algorithm ensures data stability, integration and privacy for those data files which are in size around 1 K.B to 512 K.B. Considering the lower time and power consumptions and the hardware capabilities of IoT devices, this solution provides a cheap and effective alternative for them**.**

General Terms

Light Weight Encryption, Small Scale Data, Encryption, Decryption, Security, Privacy.

Keywords

LWE, Algorithm, IoT, Security, Cryptography.

1. INTRODUCTION

The Internet of Things (IoT) has become a ubiquitous term to describe the tens of billions of devices that have sensing or actuation capabilities, and are connected to each other via the Internet [1]. The IoT includes everything from wearable fitness bands and smart home appliances to factory control devices, medical devices and even automobiles [2]. Security has not been a high priority for these devices until now. It is now time to establish The Internet of Secure Things. There has been a lot of discussion regarding the hacking of devices and systems to obtain information and data. However, just as critical are cyber-attacks against the devices themselves - attacks which take over control of the device and cause them to operate in dangerous and insecure ways. Unfortunately, many of these systems – thought to be safe – are still vulnerable. Though information security plays a pivotal role during internet communication in today's era of technology, none of researches were done on IoT devices own cryptography strategies inside itself while considering its scalability in terms of power consumption and hardware complexities [4,5]. There are various cryptography methods that provide a means for secure communication but mostly those focused on the IoT devices network security, while firewall cannot be adopted in a IoT device itself [2]. Cryptography is necessary for secure communications; it is not by itself sufficient. Cryptographic algorithms are widely used in the financial sector to ensure security in various financial transactions. For instance, the algorithms are used to ensure confidentiality and integrity of data such as a personal identification number (PIN) and an account number in automated teller machine (ATM) transactions. The application of various 10 cryptography algorithms in ATM machines have been discussed in detail by Cheng and Kurita [1,2]. The algorithms are also used to authenticate counterparties of the transactions in Internet banking services. The use of cryptography algorithms in e-banking system is one of the common topics of modern IT researchers. Some of the remarkable works done on this topic are the studies done by Srinivasan and Hiltgen [3, 4]. Generally, the algorithms used for cryptography applications are classified into two types, Asymmetric methods or public key cryptography and Symmetric methods or Symmetric key cryptography [5]. Symmetric ciphers are adopted to ensure confidentiality. To ensure integrity and authenticity, a message authentication code (MAC) based on symmetric ciphers or a digital signature based on asymmetric ciphers is adopted. This kind of protection is needed in every sphere of organizations now-a-days. Individual persons, small industries and companies also need security, privacy and protection for data communication. Moreover, the growth of IoT devices further adhere to this secrecy problem, while personal data may be compromised due to the lack of hardware functionality and software adaptability in those devices. For these areas, existing algorithms would be bulk some and redundant. But as studied, observed that there are no lightweight cryptography algorithms for these purposes. The study of Cunsolo & Distefano further justified it [5]. Lightweight algorithms which can operate on small size text and doc files with considerable reliability and validity would be enough for them. This happened due to the fact that cryptography algorithms are thought to be applied on large scale data. But the fact is that privacy, security and integrity as well as minimizing the time complexities is more important for small scale data than that is for large scale data when it requires fast cryptanalysis. This thereby signifies the importance of this present research and defines a logistic background to carry on it.

This study has found that the lack of robust lightweight cryptography algorithms often encounters problems to secure small scale data of IoT devices or the applications required less cryptography adherences. This study focuses on small scale data security. It does not deal with large scale data. The test files taken are 1-512 KB in size in cross platforms i.e. Linux (Redhat Distribution), Machintos and Windows. So this algorithm will not be efficient for large data and it is not applicable for encrypting files other than .txt and .doc extension. Individual users and small companies will be benefitted from our algorithm but it will be misleading and ineffective for large companies and organizations which needs extreme privacy, security and deal with huge amounts of data at a time.

2. LITERATURE REVIEW

Recent advancement of communication and computing technologies introduces different types of portable devices that populate in present day to day life. These devices have limited battery power, restricted storage and low computation power to bring the device in affordable cost and portable size. Information security is of primary concern for all users irrespective of the computing device being used. Among the different approaches for achieving information security, the present work concerns with data cryptography techniques for the small scaled devices i.e IoT devices [6,7-9].

The Internet of Things is comprised of a wildly diverse range of device types- from small to large, from simple to complex – from consumer gadgets to sophisticated systems found in DoD, utility and industrial/manufacturing systems. For over 25 years, cybersecurity has been a critical focus for large enterprises, whereas it has only recently become a focus for most engineers building embedded computing devices. “Experience is the best teacher, but the tuition is high”, or so goes the saying [15]. Rather than learn all the lessons by experience, embedded engineers can take a page from the enterprise security playbook. What are the challenges for implementing the Internet of Secure Things and assuring security of embedded devices? The specialized nature of these devices presents the following challenges [12-17]: (a) Critical functionality (b) Replication (c) Security assumptions (d) Not easily patched (e) Long life cycle but short battery storage capacity for simulating large task (f) Proprietary/industry specific protocols (g) Deployed outside of enterprise security perimeter. Varieties of encryption algorithms are available to encrypt the data but execution of the traditional encryption algorithms consumes time, space and energy. Moreover, side channel attacks are based on time and power that can be applied to the block ciphers implemented on smart card technology. Also protecting implementation against these kinds of attacks is usually difficult and expensive.

Application of cryptographically strong algorithm such as AES-Rijndael leads to significant transmission delay, and require high computations as well as large storage capacity. It leads to infeasible to incorporate the strong cryptographic algorithms in the resource constrained devices [16]. For these reasons, researchers are becoming more interested to lightweight cryptography day by day. Many development initiatives have been taken on lightweight symmetric key algorithms to make security process simpler, more integrated and less time and resource consuming. Lightweight cryptographies have been implemented successfully in several fields to ensure security. For example- Kumar & Aggarwal have developed a lightweight cryptography solution for resource constraint mobile ad-hoc networks (MANET), Barbero & Ytrehus have developed lightweight cryptography for RFID devices [17, 18]. Lightweight cryptography scheme has also been used in neural network [19]. On the other hand, encryption algorithm ICEBERG proposed for its implementation with special emphasis to the reconfiguration hardware. However, the software implementation is not suitable i.e., not cost effective with respect to storage requirement and/ or speed. Data dependent permutation (DDP)-based fast encryption algorithms are appeared to be faster and efficient for high speed networks. Recently, Cobra-H64 and Cobra-H128 were proposed in use switchable operations to prevent the weak keys identified against the earlier DDP-based encryption techniques. These ciphers are specially emphasized for high speed performance hardware implementation but require more hardware resources [20]. Efficient for high speed networks. Recently, Cobra-H64 and Cobra-H128 were proposed in use switchable operations to prevent the weak keys identified against the earlier DDP-based encryption techniques. These ciphers are specially emphasized for high speed performance hardware implementation but require more hardware resources [20].

Lightweight cryptography protocol was further tailored for implementation in constrained environments including RFID tags, sensors, contactless smart cards, health-care devices and so on. The properties of lightweight cryptography have already been discussed in ISO/IEC 29192 in ISO/IEC JTC 1/SC 27. ISO/IEC 29192 is a new standardization project of lightweight cryptography, and the project is in process of standardization [22].

In ISO/IEC 29192, lightweight properties are described based on target platforms. In hardware implementations, chip size and/or energy consumption are the important measures to evaluate the lightweight properties. In software implementations, the smaller code and/or RAM size are preferable for the lightweight applications. From the view of the implementation properties, the lightweight primitives are superior to conventional cryptographic ones, which are currently used in the Internet security protocols, e.g., IPsec, TLS. Lightweight cryptography also delivers adequate security. Lightweight cryptography does not always exploit the security-efficiency trade-offs. Report recent technologies of lightweight cryptographic primitives.

As the scale of the data used and time and space complexities of lightweight encryption are small, it can’t be used for large levels of data. But surely it can fasten the secure information transaction process in the educational and organizational sector. Here the main challenge is increasing the efficiency and lowering the error percentage. So researchers focus on this topic and try to make that optimum balance among cost, efficiency and performance while considering the device battery life for cryptanalysis.

3. PROPOSED MODEL OF LWC

3.1 Random key assignment

A four digit random key like abcd, efgh, ijkl etc is taken. Four digit key is chosen because it is neither too short nor too long. Too short key length would be easy to crack and too long key length will lower the speed of the algorithm. Another reason for choosing four letter key is that most PIN numbers, security codes are four digits. The key is changed during every iteration and new key is generated. This makes the algorithm hard to crack. The number of iterations of the algorithm is equal to the number of letters in the sentence including space and special characters.

3.2 Key encryption and new key generation

The ASCII values of the key letters are calculated and then summed up. Then the mod of the sum by 100 is calculated. The result is converted into 8-bit binary number system and divided into four blocks that is each block contains two bits.

The two bits of each block are concatenated with three binary bits ranging from 000 to 111 from both directions i.e. forward and backward. In first iteration, the concatenation is done with 000, in the second iteration it is done with 001 and so on. After eight iterations, it again comes back to 000. So now each block contains eight digits. The decimal values of these new blocks are found out and these are the ASCII values of new key. Thus new key is generated from previous one. This procedure continues up to the last iteration. 3.3.3 Small Scale Data Encryption and Decryption In the encryption part, first the ASCII value of the plaintext is found out. Then, its corresponding binary value is calculated. Then, take a random number 298 which is fixed for this algorithm. The mod of 298 is done by the value which was found when the sum of the ASCII values of the key letters was mod by 100. The result is then converted into binary and then XOR-ed with the binary value of the ASCII value of the plaintext. The result of the XOR is again converted to decimal and this decimal value is the ASCII value of the ciphertext. The character corresponding to that ASCII value is found out and this is the ciphertext of the plaintext. Thus encryption is done. Similarly, in the decryption part, first the ASCII value of the ciphertext is found out and then its corresponding binary value is found out. Again, the mod of random number 298 is done by the value which was found when the sum of the ASCII values of the key letters was mod by 100. This value was transmitted to the decryption part. This new result is converted into binary and XOR-ed with the binary value of the ASCII value of the ciphertext. The result of the XOR is again converted to decimal and this decimal value is the ASCII value of the plaintext. The character corresponding to that ASCII value is found out and this is the plaintext of the ciphertext. In this way, decryption is done. To make the whole process more clear and visible to the reader, an example is taken The sentence “MY MIST ” is taken as the plaintext. As it is a symmetric key algorithm, the key for encryption and decryption will be same. The string “abcd ”is taken as the random key. The letter by letter encryption and decryption technique of the sentence is shown in the following figures.

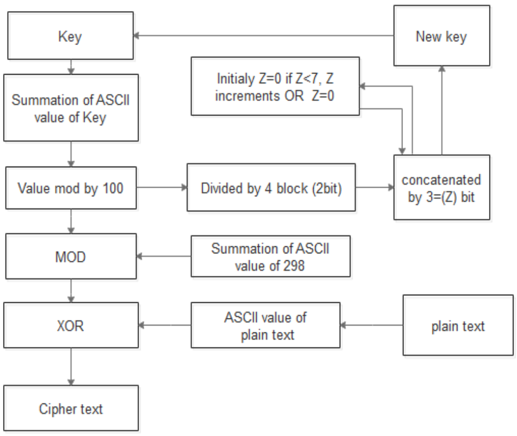

Key
New key
Summation of ASCII value of Key
Initialy Z=0 if Z<7, Z increments OR Z=0
Value mod by 100
Divided by 4 block (2bit)
concatenated by 3=(Z) bit
MOD
Summation of ASCII value of 298
XOR
ASCII value of plain text
plain text
Cipher text

Fig 1: Encryption Flowchart

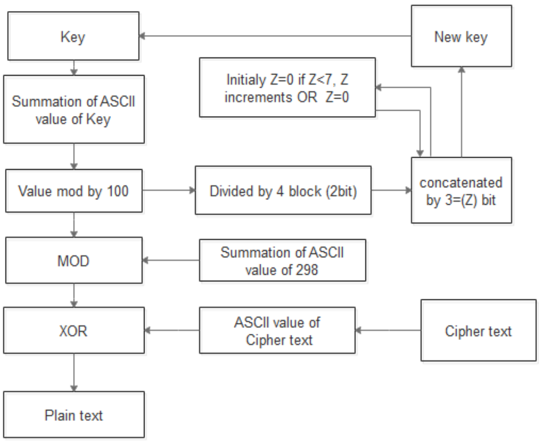

Key
New key
Initialy Z=0 if Z<7, Z increments OR Z=0
Summation of ASCII value of Key
Value mod by 100
Divided by 4 block (2bit)
concatenated by 3=(Z) bit
MOD
Summation of ASCII value of 298
XOR
ASCII value of Cipher text
Cipher text
Plain text

Fig 2: Decryption Flowchart

4. THE LWC ALGORITHM

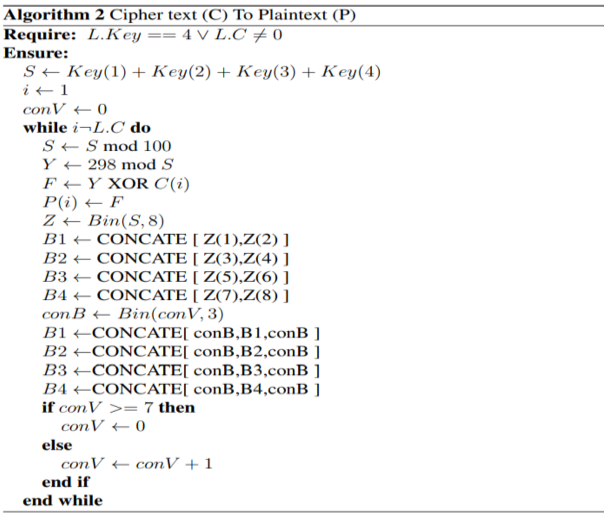
4.1 Encryption Algorithm

In Algorithm 1 show how to encrypt a plaintext data to cipher text data.

```
Algorithm 1 Plaintext (P) To Cipher text (C)
Require: $L.Key == 4 \vee L.P \neq 0$
Ensure:
  $S \leftarrow Key(1) + Key(2) + Key(3) + Key(4)$
  $i \leftarrow 1$
  $conV \leftarrow 0$
  while $i \neg L.P$ do
    $S \leftarrow S$ mod 100
    $Y \leftarrow$ 298 mod $S$
    $F \leftarrow Y$ XOR $P(i)$
    $C(i) \leftarrow F$
    $Z \leftarrow Bin(S, 8)$
    $B1 \leftarrow$ CONCATE [ Z(1),Z(2) ]
    $B2 \leftarrow$ CONCATE [ Z(3),Z(4) ]
    $B3 \leftarrow$ CONCATE [ Z(5),Z(6) ]
    $B4 \leftarrow$ CONCATE [ Z(7),Z(8) ]
    $conB \leftarrow Bin(conV, 3)$
    $B1 \leftarrow$CONCATE[ conB,B1,conB ]
    $B2 \leftarrow$CONCATE[ conB,B2,conB ]
    $B3 \leftarrow$CONCATE[ conB,B3,conB ]
    $B4 \leftarrow$CONCATE[ conB,B4,conB ]
    if $conV >= 7$ then
      $conV \leftarrow 0$
    else
      $conV \leftarrow conV + 1$
    end if
  end while
```

4.2 Decryption Algorithm

In Algorithm 2 show how to decrypt a cipher text data to plain text data.



```
Algorithm 2 Cipher text (C) To Plaintext (P)
Require: $L.Key == 4 \vee L.C \neq 0$
Ensure:
  $S \leftarrow Key(1) + Key(2) + Key(3) + Key(4)$
  $i \leftarrow 1$
  $conV \leftarrow 0$
  while $i \neg L.C$ do
    $S \leftarrow S$ mod 100
    $Y \leftarrow 298$ mod $S$
    $F \leftarrow Y$ XOR $C(i)$
    $P(i) \leftarrow F$
    $Z \leftarrow Bin(S, 8)$
    $B1 \leftarrow$ CONCATE [ Z(1),Z(2) ]
    $B2 \leftarrow$ CONCATE [ Z(3),Z(4) ]
    $B3 \leftarrow$ CONCATE [ Z(5),Z(6) ]
    $B4 \leftarrow$ CONCATE [ Z(7),Z(8) ]
    $conB \leftarrow Bin(conV, 3)$
    $B1 \leftarrow$CONCATE[ conB,B1,conB ]
    $B2 \leftarrow$CONCATE[ conB,B2,conB ]
    $B3 \leftarrow$CONCATE[ conB,B3,conB ]
    $B4 \leftarrow$CONCATE[ conB,B4,conB ]
    if $conV >= 7$ then
      $conV \leftarrow 0$
    else
      $conV \leftarrow conV + 1$
    end if
  end while
```

4.3 Sampling Strategy

The data into seven main categories have been categories: only characters, only numbers, only special characters, char-special char, char-number, num-special character, char-numspecial character. 300 files of each category has been taken. So there are total 2800 (300*7) test files. The files are 1-512 KB in size as our algorithm will deal only with small scale data. The file types are .txt and .doc. Random sampling have been used in data. Random samples are used in population sampling situations when reviewing historical or batch data. 45 The key to random sampling is that each unit in the population has an equal probability of being selected in the sample. Using random sampling protects against bias being introduced in the sampling process, and hence, it helps in obtaining a representative sample. In general, random samples are taken by assigning a number to each unit in the population and using a random number table or Minitab to generate the sample list. Absent knowledge about the factors for stratification for a population, a random sample is a useful first step in obtaining samples. For example, an improvement team in a human resources department wanted an accurate estimate of what proportion of employees had completed a personal development plan and reviewed it with their managers. The team used its database to obtain a list of all associates. Each associate on the list was assigned a number. Statistical software was used to generate a list of numbers to be sampled, and an estimate was made from the sample. The files have run on different platforms like windows, Linux, Mac. So our research and analysis is platform independent.

5. EXPERIMENTAL ANALYSIS AND RESULT

For the simulation of lightweight symmetric key cryptography algorithm Mathlab had been used. For result analysis, 1 K.B to 512 K.B range text file and 1 K.B to 66.5 K.B doc type files had been encrypted and decrypted and its execution time had been collected (see Appendix). There were 7 categories of text files around 1 K.B to 512 K.B and doc type files around 1 K.B to 66.5 K.B had been executed on three different platforms like Windows, Linux and Mac OS and the categories of files were : (a) Plain character data text file and doc type file. (b) Plain number data text file and doc type file. (c) Plain special character data text file and doc type file. (d) Plain character data and number data text file and doc type file. (e) Plain special character data and number data text file and doc type file. (f) Plain character data and special character data text file and doc type file. (g) Plain character data, special character data and number data text file and doc type file.

Table 1. Analysis on Category 1 Only plain Character data

File category	**File Type**	**Used Plat-form**	**Used key length**	**Average iteration**	**Per iteration Execution Time**	**Total Execution time (sec)**
Category - 1(150K.B for txt 20K.B for doc type)	.txt	Windows	4	164, 410	1232	133.393
	.doc/.docx	Windows	4	164, 410	1215	135.252
	.txt	Linux	4	164, 410	1213	135.459
	.odt/.fodt/.uot	Linux	4	164, 410	1213	135.323
	.txt	Mac-OS	4	164, 410	1204	136.489
	.page	Mac OS	4	164, 410	1206	134.256
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

Table 2. Analysis on Category 2 Only plain Character data

File category	**File Type**	**Used Plat-form**	**Used key length**	**Average iteration**	**Per iteration Execution Time**	**Total Execution time (sec)**
Category-2(150K.B for txt 20K.B for doc type)	.txt	Windows	4	115393	902	127.8763
	.doc/.docx	Windows	4	115393	902	127.1125
	.txt	Linux	4	115393	887	129.97856
	.odt/.fodt/.uot	Linux	4	115393	886	128.555
	.txt	Mac-OS	4	115393	891	129.48796
	.page	Mac OS	4	115393	890	128.1235
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

Table 3. Analysis on Category 3 Only plain Character data

File category	File Type	Used Plat-form	Used key length	Average iteration	Per iteration Execution Time	Total Execution time (sec)
Category-3(150K.B for txt 20K.B for doc type)	.txt	Windows	4	115393	902	127
	.doc/.docx	Windows	4	115393	902	127.19755
	.txt	Linux	4	115393	887	129.125
	.odt/.fodt/.uot	Linux	4	115393	886	128.578
	.txt	Mac-OS	4	115393	891	129.90
	.page	Mac OS	4	115393	890	128.8934
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

Table 4. Analysis on Category 4 Only plain Character data

File category	File Type	Used Plat-form	Used key length	Average iteration	Per iteration Execution Time	Total Execution time (sec)
Category-4(150K.B for txt 20K.B for doc type)	.txt	Windows	4	116504	902	127.8763
	.doc/.docx	Windows	4	116504	902	127.1125
	.txt	Linux	4	116504	887	129.97856
	.odt/.fodt/.uot	Linux	4	116504	886	128.555
	.txt	Mac-OS	4	116504	891	129.48796
	.page	Mac OS	4	116504	890	128.1235
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

Table 5. Analysis on Category 5 Only plain Character data

File category	File Type	Used Plat-form	Used key length	Average iteration	Per iteration Execution Time	Total Execution time (sec)
Category-5(150K.B for txt 20K.B for doc type)	.txt	Windows	4	116504	980	127
	.doc/.docx	Windows	4	109405	972	133.0
	.txt	Linux	4	109405	969	132.90
	.odt/.fodt/.uot	Linux	4	109405	969	131.157
	.txt	Mac-OS	4	109405	964	131.90
	.page	Mac OS	4	109405	963	130.34
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

Table 6. Analysis on Category 6 Only plain Character data

File category	File Type	Used Plat-form	Used key length	Average iteration	Per iteration Execution Time	Total Execution time (sec)
Category-6(150K.B for txt 20K.B for doc type)	.txt	Windows	4	164.410	1232	133.393
	.doc/.docx	Windows	4	164.410	1215	135.252
	.txt	Linux	4	164.410	1213	135.459
	.odt/.fodt/.uot	Linux	4	164.410	1213	135.323
	.txt	Mac-OS	4	164.410	1204	136.489
	.page	Mac OS	4	164.410	1206	134.256
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

Table 7. Analysis on Category 7 Only plain Character data

File category	File Type	Used Platform	Used key length	Average iteration	Per iteration Execution Time	Total Execution time (sec)
Category-7(150K.B for txt 20K.B for doc type)	.txt	Windows	4	151,912	1201	130.9
	.doc/ .docx	Windows	4	151,912	1196	130.0
	.txt	Linux	4	151,912	1195	129.8
	.odt/. fodt/. uot	Linux	4	151,912	1193	128.0
	.txt	Mac-OS	4	151,912	1194	128.1
	.page	Mac OS	4	151,912	1193	127.5
	Txt file equivalent type .txt and docfile equivalent type .doc, .docx, .odt, .fodt and .out					

The proposed lightweight symmetric key cryptography algorithm (LWE) has used on 500 text file as well as doc type files where sizes of the files were 1 K.B to 512 K.B. There were three different types of platforms used for this purpose. The platform that had been used for the proposed lightweight symmetric key cryptography algorithm were Linux, widows, Mac OS in order to find out the efficiency of the algorithm in different platform and to define its platform independent so that any user of any platform could be benefitted using 57 this algorithm. Their average iteration, per iteration/execution time, total execution time had been calculated in order to find out efficiency, the problem and the effectiveness of the algorithm on security. This proposed lightweight symmetric key cryptography algorithm provided security on seven types of text files as well as the doc types files. They were Plain character data text file and doc type file, Plain number data text file and doc type file, Plain special character data text file and doc type file, Plain character data and number data text file and doc type file, Plain special character data and number data text file and doc type file, Plain character data and special character data text file and doc type file, Plain character data, special character data and number data text file and doc type file where the sizes of text files were around 1 K.B to 512 K.B as well as the doc type files were around 1 K.B to 66.5 K.B. As a result, the encryption and decryption process of the algorithm was less time consuming, less complex as well as little percentage of error. Because of this, LWC algorithm was easy to implement and also provide better security during data transmission. The algorithm ensures data stability, integration and privacy. It is not feasible for small industries to spend large amount of money for buying costly antivirus soft-wares and other security measures. Our algorithm provides a cheap and effective alternative for them. The algorithm had been tested on various platforms and with various categories of input data which makes it very reliable and valid. So the user could have valid output at a low price. In result analysis, the lightweight symmetric key cryptography algorithm was executed on different platforms (Windows, Linux, Mac OS) for 150 K.B text file and for 20 K.B for type files showed and discussed using table. It also discussed about file category, file type, used platform, used key length, average iteration, per iteration/Execution time and total Execution time on seven categories in three different platforms.

6. CONCLUSION

This LWC algorithm provides an easy alternative for them to protect their data efficiently at a lower time complexity, which thereby suitable for the IoT devices to adopt it a feasible solution for cryptanalysis. The proposed algorithm has been implemented in three main stages- Random Key Assignment, Key Encryption and New Key generation and finally Small Scale Data Encryption and Decryption. Though this algorithm encrypts only English characters, special characters, numbers and their combinations. It is not applicable for other languages. Future studies may include other languages also by modifying the algorithm for UNICODE characters. As the study was limited to a specific time frame, the robustness of the LWC algorithm could not be assessed in comparison with other existing encryption algorithms.

However, this study proposes a guideline for future light weight cryptology on pdf, .db and .jpg file extensions. The size of the text files may also vary from 512 KB for image, .pdf, media and .db files. To encrypt image files with this LWC algorithm, first the image has to be converted into matrix. Then a pattern has to be generated within the matrix. To generate the pattern, future researcher need a key. So for the modified algorithm for image encryption and decryption, two different keys are needed. One key is for pattern generation and another for encryption and decryption. This research can act as a guideline for further researches which will enhance the lightweight encryption system in a far greater way and increase the performance and scope manifolds. With a further modification, the proposed LWC algorithm can be used for IoT devices where power consumption and hardware limitations are major factors.

7. REFERENCES

[1] W. Wanping, J. Jin, and J. Cheng, “The research and design of ATM PIN pad based on triple DES,” in IEEE International Conference on Information and Automation (ICIA), (Shenzhen), pp. 443 – 447, IEEE, 2011.

[2] S. Kurita and K. Komoriya, “Privacy protection on transfer system of Automated Teller Machine from brute force attack,” in International Conference on Advanced Information Networking and Applications Workshops (WAINA), (Fukuoka), pp. 72 – 77, IEEE, 2012.

[3] O. Dandash, P. D. Le, and B. Srinivasan, “Security analysis for internet banking models,” in Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, (Qingdao), pp. 1141 – 1146, IEEE, 2007.

[4] A. Hiltgen, T. Kramp, and T. Weigold, “Secure internet banking authentication,” Security & Privacy, IEEE, pp. 21 – 29, 2006.

[5] V. Cunsolo and S. Distefano, “Achieving information security in network computing systems,” in Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, (Chengdu), pp. 71 – 77, IEEE, 2009.

[6] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, “Lightweight cryptography for embedded systems a comparative analysis,” (Egham), 6th

International Workshop on Autonomous and Spontaneous Security, 2013.

[7] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: ultra-lightweight cryptography for resource-constrained devices," in FC'10 Proceedings of the 14th international conference on Financial cryptograpy and data security, (Heidelberg), pp. 3–18, Springer-Verlag Berlin, 2010.

[8] O. Goldreich, Foundations of Cryptography. Cambridge: Cambridge university press, 2004.

[9] H. Delfs and H. Knebl, Introduction to cryptography: principles and applications. Nurnberg: Springer Publications, 2007.

[10] L. A. Baglione, "Writing a research paper in political science: A practical guide to inquiry, structure, and methods," Thousand Oaks: CQ Press, 2nd edition ed., 2012.

[11] R. M. Nieswiadomy, Foundations of Nursing Research. Michigan: Appleton & Lange, 1998.

[12] J. H. Geri LoBiondo-Wood, Nursing Research: Methods and Critical Appraisal for Evidence-Based Practice, (Nursing Research: Methods, Critical Appraisal & Utilization). Sydney: Libby Houston and Melinda McEvoy, 8 th ed., 2013.

[13] C. Shannon, "Communication theory of secrecy systems," 1949.

[14] B. Schneier, "Applied cryptography: Protocols, algorithms, and source code in c," New Jersey: John Wiley & Sons, 1996.

[15] S. Knapskog, "New cryptographic primitives," in Computer Information Systems and Industrial Management Applications (CISIM) Conference, (Ostrava), pp. 3 – 7, IEEE, 2008.

[16] A. Kumar, K. Gopal, and A. Aggarwal, "A complete, efficient and lightweight cryptography solution for resource contrainst mobile ad-hoc networks," in 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC), (Solan), pp. 854 – 860, IEEE, 2012.

[17] A. Barbero, G. Horler, A. Kholosha, and O. Ytrehus, "Lightweight cryptography for rfid devices," in IET International Conference on Wireless, Mobile and Multimedia Networks, (Beijing), pp. 294 – 297, IET, 2008.

[18] M. Stottinger, S. Huss, S. Muhlbach, and A. Koch, "Side-channel resistance evaluation of a neural network based lightweight cryptography scheme," in IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), (Hong Kong), pp. 603 – 608, IEEE, 2010.

[19] S. Tripathy and S. Nandi, "Lcase: Lightweight cellular automata-based symmetric-key encryption," International Journal of Network Security, p. 243, 2009.

[20] D. Bradbury, "feature: Encryption-the-key-to-secure-data: computerweekly.," April 2005. Retrieved December 20, 2014, from computerweekly: http://www.computerweekly.com/feature/Encryption-the-key-to-secure-data.

[21] M. Katagi and S. Moriai, "Lightweight cryptography for the internet of things," Tokyo: Sony Corporation, 2010.

[22] https://www.iconlabs.com/prod/internet-secure-things-%E2%80%93-what-really-needed-secure-internet-things