Evaluation of symmetric key algorithms for Body Sensor Networks (BSN) on Raspberry pi

Akash Chauhan*, Kanika Mathur*, Divyashikha Sethia*

*Department of Computer Engineering, Delhi Technological University, Main Bawana Road, Delhi, India

[1]akash75457@gmail.com, [2]mathurkanika1@gmail.com, [3]divyashikha@dce.edu

***Abstract*- BSNs (Body sensor networks) have been widely implemented in medical environments. Monitoring the patient health and helping in fast and timely data access and transfer, BSNs pose a major reform in the field of medicine. There have been various techniques through which BSNs have been implemented, one being that using the Raspberry pi, a single board computer (SBC), which presented a new milestone in the field of sensor networks [10]. The data being sent over these sensor devices is critical and with increasing software attacks, comes the major challenge of providing a secure data transmission. The BSN implementation using the Raspberry pi incorporates the RC 4 encryption to establish a secure communication. But situations have been witnessed where RC4 does not suffice to the needs of the situation and in fact has detrimental overheads on the system.**

In this paper, various symmetric key cryptography algorithms used in the BSNs have been segregated namely RC4, RC5, Blowfish, and Skipjack. An implementation of these algorithms has been carried out on the Raspberry pi. Further using available LINUX packages/utilities, a comparison of these algorithms has been laid forth as to find the most suited algorithm in a given situation. Also by simulating the values obtained for the algorithms on MATLAB, a proper graphical analysis of the entire work has been depicted. A situational use of these algorithms along with their weaknesses given will help in segregating the use of these algorithms according to the need of the time and situation.

***Index Terms*—BSN, encryption, Blowfish, RC4, RC5, Skipjack, Single board computer, Raspberry pi, Accelerometer, LINUX package/utilities**

I. INTRODUCTION

With the ever increasing incorporation of technology into healthcare, and with advances being made regularly, one can imagine the future time nursing homes or hospitals running on pervasive networks that can not only provide a continuous medical monitoring , but also a medical data access and also emergency communication. A BSN is a wireless ad hoc network that has sensors that are attached to patient body and also medical devices kept in close proximity. Various implementations of BSNs have been proposed. One of them is using the Raspberry pi [10]. As the case of sensor devices, data being sent and accessed remotely, any tampering with patient statistics can lead to dangerous outcomes. Thus data security becomes an issue of prime concern. Thus we need an efficient crypto system so as to securely transmit data.

Various algorithms such as RC4 [4], RC5 [2] are popular for such use. The use of an algorithm for BSN is critical as these devices have constraints such as low processing power, less memory and limited bandwidth. In the implementation of a BSN using the raspberry pi, the entire model was made to use RC 4 for data encryption. But in not all situations, do we find RC 4 to be the best suitable algorithm. There can be situations where other algorithms work better.

In our work, various symmetric key encryption algorithms used in BSNs have been segregated. The reason behind using symmetric key algorithms is that since BSNs are low computation devices, they need algorithms involving less amount of computation work, which is provided by these algorithms. Further a comparison concerning various factors of performance such as execution time, CPU usage, Cache miss rate, Page faults has been carried out. Later discussing the issues relating to the algorithms and determining situations where these algorithms suite best, we present what is called situational use of the algorithms.

The paper is broadly divided into 7 sections. Section I provides an introduction to the body sensor networks and our work. Section II includes the literature survey about various algorithms used in BSN. The implementation task is described in Section III, with the proceeding section IV depicting the results of comparison. Section V contains the issues encountered with the various encryption algorithms. With Section VI, we present a situation based use of various algorithms today. We conclude with Section VII, thus summarizing our work with a tabular comparison of the performance of algorithms.

II. LITERATURE SURVEY ON SECURITY

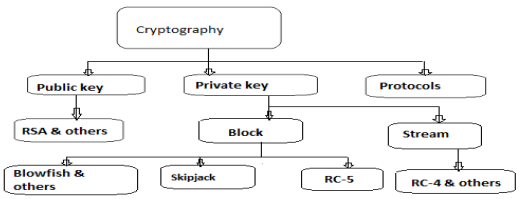


Figure 1: A hierarchical tree depicting types of cryptography algorithms

A survey was conducted in the data security field to find out various cryptography algorithms used in sensor devices and specifically those used in BSN.

Public key encryption methods have not been much successful on bio sensor nodes as these devices have a low computation power [7]. IBE-Lite is also not so suitable for BSN since no proper authentication is provided and it is susceptible to node duplication attacks [7]. Despite its use in BSN's today, Public key encryption methods are not so desirable for the BSN as separate keys are required which makes their implementation difficult on these low power and computation devices. Moving to Private Key encryption (symmetric key), it is widely used for BSN; however symmetric key ciphers can also be expensive to implement on some target platforms. Private key cipher(-divided into Stream and Block Cipher), Stream ciphers

(RC 4, RC5 etc.) have a simple architecture and a fast encryption rate and are thus considered more suitable for sensors having limited memory and computing resources as compared to block ciphers (AES, DES etc) [3].

Various research works indicate that the most commonly used algorithm for BSN is RC-5. According to the Gawali and Wadhai 2012, the RC5 Algorithm [2] is good choice on the basis of its overall performance [2]. Amini, Verhoeven, Lukkien and Chen 2011, considered that the RC4 algorithm [4] is a good option to use for small size messages in BSN. RC5 can be considered as one of the best ciphers in terms of overall performance, when used in nodes with limited memory and processing capabilities [2]. But various tradeoffs to this exist. Situations having message/input data not large enough, use RC4 [4], whereas those needing block encryption, use RC5 [2]. Also other algorithms include DES, Triple DES [5], AES [2], Blowfish, Skipjack, SHA 1(secure hash algorithm), MD5 (for message digests), A5, RC-4 [4] and RC- 6. AES [2] is also widespread, having inbuilt hardware support for some platforms. Taking the energy consumption of AES, RC-5 and DES, we find RC-5 to consume lower energy [2]. RC5 is better than DES in security strength and implementation efficiency [2]. RC-4 and Skipjack can be used to provide lightweight message confidentiality for our security system [4]. Elminaam, Kader, and Hadhoud 2010, conclude that Blowfish [5] has a comparatively better performance than AES [2], DES [2], Triple DES [5], RC6 and RC2. Tze Huei Lai, Begg, Palaniswami in publication-Healthcare sensor networks challenges toward practical implementation stated that Skipjack algorithm is also used in BSN [9].

III. IMPLEMENTATIONS

A. *On the Raspberry pi*

The Accelerometer senses the readings of the Pi, which are accessed by the pi using python scripts [10]. After the execution of this script on the pi, the data is stored in certain text files out of which three have been considered (10, 100 and 1000 accelerometer readings). Various algorithms have first been implemented in C language. The performance of individual algorithms is then stored in text files for different size of input data and key size. Furthermore the various factor comparisons, results and statistics have been simulated on matlab to present a graphical depiction. The algorithms implemented include RC-4 (a stream cipher), RC-5 (a symmetric key block cipher), Blowfish (a block cipher), Skipjack (a block cipher).

B. *Implementation of comparison factors:*

The implementation of the entire comparison task has been carried out using various kernel level packages and commands for all algorithms, separately varying the Accelerometer Input (from 100 to 2000) for each algorithm. The description of the implementation of the each factor has been given below.

a) Execution Time:

C language function named clock () has been used to measure the current time of the system clock (before and after the program execution). The difference in values of two clock () functions gives execution time for a program.

b) Memory occupied at Run Time:

Linux based utility has been used to measure the amount of memory occupied at run time for a program. The "size" command is used thus giving the final output in bytes. Memory Consumed (MC) at run time does not vary with the Accelerometer Input.

c) Lines of code:

Linux based utility have been used to find out the number of lines in a program. The "nl" command gives information about how many lines of code a program contains.

d) Context Switches per second (CSS)

Once again Linux based utility commands have been used to find out the number of Context Switches occurring per Second in a program. Colin King, a well known kernel Engineer, first introduced the linux based package named health-check [22]. A reference has also been taken from git clone [24] to install this package on the Raspberry pi. Before the installation of this package, another package named libjson0-dev was installed. Thus the sequence of package installation is libjson0-dev first and then health-check.

e) Page Faults per Second:

Linux based utility has been used to find out the number of Page Faults occurring per Second for a program. The procedure used is same as that for context switches occurring per second.

f) CPU usage and time:

We have used Linux based utility to find out CPU usage by a program. The procedure used is same as that for context switches occurring per second. Similarly we can obtain the CPU time for a particular algorithm using the same procedure.

g) Cache Miss Rate:

Linux based utility has been used to find out the Cache Miss Rate (CMR) for a program. For this valgrind linux based package has to be installed. This gives us result involving the usage of Cache Miss Rate (CMR) including all type of caches like I1, LLi, D1, LLD and LL.

h) Read Operations per Second:

We have used Linux based utility to find out number of Read Operations occur per Second for a program. The procedure used is the same as that for Context switches occurring per second.

i) Write Operations per Second:

Linux based utility has been used to find out the number of Write Operations occurring per Second for a program. The procedure used is same as is in context switch occurring per seconds.

j) Power Consumption:

Linux based utilities are used to find out how much power is required by a program. We tried to use the packages introduced by Colin King, namely power-calibrate and health-check on the Raspberry Pi [22] [23]. But unfortunately the power-calibrate package did not work out on the Raspberry pi.

The power-calibrate package (measures the amount of battery power a program consumes) actually requires Direct Current (DC) supply. All possible efforts had been tried to give Direct Current (DC) supply to the Raspberry pi. When using the portable USB bank to provide the DC supply, we observed the Raspberry pi operating nicely but the power-calibrate package still not working. Thus all these steps to add external DC source went in vain.

So another way was devised to help run the power calibrate package. We used Ubuntu to calculate the power consumption. The laptop battery was used to give Direct Current (DC) and thus facilitate the working of power calibrate package. Reference from git clone [24] was used to install this package on Ubuntu. After the installation of this package the power consumed to use just 1% of the CPU and power consumed in carrying out 1 context switch, on the machine was obtained. The package works only when the machine is running in Direct Current (DC). Along with this the health-check package was also installed. The remaining procedure is the same as in case of calculating context switches occurring per second.

Further on running the health-check package, the number of Context Switches occurring per second and total CPU usage was obtained. After simple multiplications, power consumption for a program is calculated. Doing the same for all algorithms and for 100 to 2000 Accelerometer Input, we get the power consumption for all algorithms.

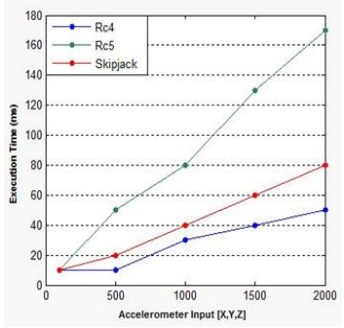
IV. TESTING AND RESULTS

Assumption: The Accelerometer readings vary from 100 to 2000 Input entries. Each individual input of the Accelerometer is a depiction of the position of a person (in 3D co-ordinate system).

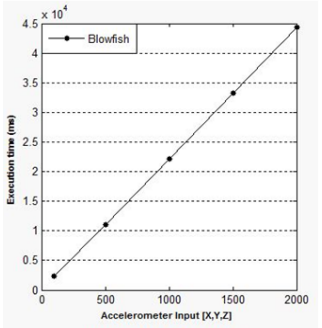
A. Execution time and CPU time

Graphs 1 represents Execution times verses Accelerometer input for RC4, RC5 and Skipjack; and graph 2 represents Execution times verses Accelerometer input for Blowfish. As is eminent from the graph 1 and 2, Blowfish takes a time that is far more than the rest of the algorithms while RC4 requires the least time (for all sizes of Input data.). Similar results are valid for CPU time as Execution time is directly proportional to CPU time.

In Body Sensor Network (BSN), where the time consumption is a major constraint, we find RC4 to be the best suitable choice.

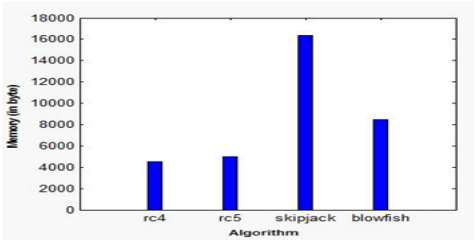


Graph 1



Graph 2

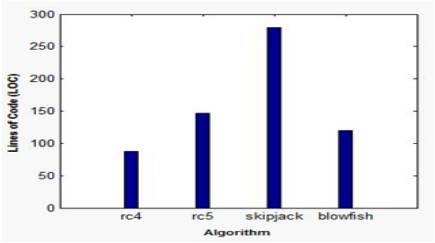
B. Memory occupied at runtime



Graph 3: Memory occupied verses algorithm

As is seen in graph 3, Skipjack algorithm uses a larger amount of memory space as compared to the rest of the algorithms; whereas RC4 occupies minimum memory at run time. In BSN where memory management is necessary, we can say RC4 is a good choice.

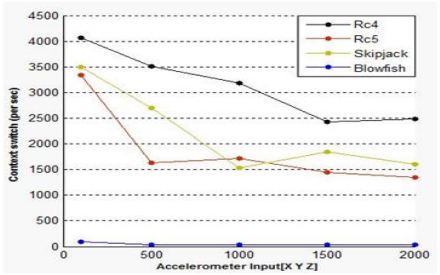
C. LOC (Lines of Code)



Graph 4: Lines of code verses algorithm

As graph 4 shows, Skipjack has maximum Lines of Code (LOC) than all the other Algorithms and RC4 has the minimum. Since BSN have less storage space, less number of Lines of Code (LOC) is preferred, which justifies RC4 to be a good choice.

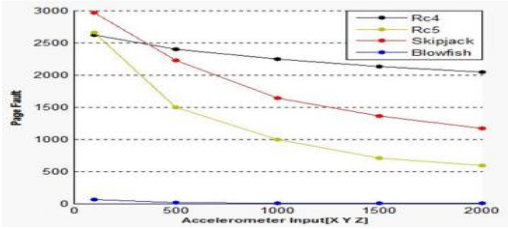
D. Context switches per second



Graph 5: Context switches per sec verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

We conclude from the graph 5 that maximum Context Switches per Second (CSS) occur for RC4 algorithm whereas Minimum Context Switches per Second (CSS) occur for Blowfish. So in Body Sensor Network (BSN), we can say that Blowfish algorithm is a good choice.

E. Page faults per second

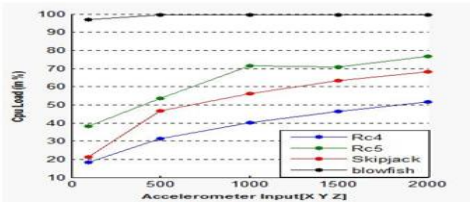


Graph 6: Page faults verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

As we can see in graph 6 above, minimum Page Faults per Second (PFS) occur for Blowfish algorithm considering all sizes of input and maximum Page Faults per Second (PFS)

occur for skipjack algorithm. But as value of inputs increase, we find the maximum Page Faults per Second (PFS) occurring in case of RC4 algorithm. Thus, as far as Page Faults per Second (PFS) are concerned, we conclude that Blowfish Algorithm is the best suitable algorithm. RC4 is not suitable when the amount of input data becomes large.

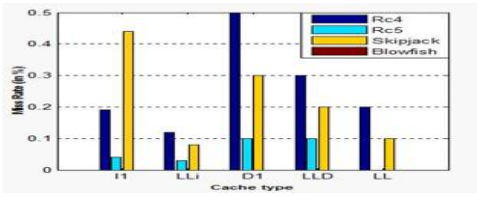
F. CPU Usage



Graph 7: CPU Load verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

We observe that the maximum load on CPU is incurred in case of Blowfish algorithm and the minimum load is for RC4 algorithm for all sizes of input data. In all the cases where CPU load is a constraint, we conclude the use of RC4 algorithm to be the best.

G. Cache Miss Rate

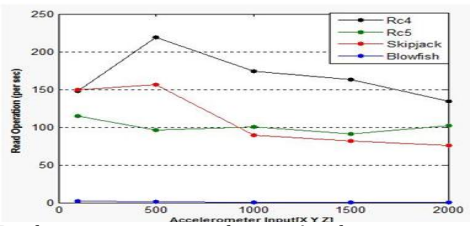


Graph 8: Cache miss rate verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

Here we focus on all the different kinds of caches. Considering the statistics of I1 cache, the maximum Cache Miss Rate (CMR) is for Skipjack algorithm and minimum for Blowfish Algorithm which is closely followed by RC5 Algorithm. For the case of LLi cache, we observe the maximum CMR for RC4 Algorithm and minimum for Blowfish Algorithm which is succeeded by RC5 Algorithm. For D1 cache, the maximum CMR is for RC4 Algorithm and minimum for Blowfish Algorithm following which is RC5 algorithm. If we consider LLD cache, we find the maximum CMR to be for RC4 Algorithm and minimum for Blowfish Algorithm after which is the RC5 Algorithm. For the statistics of LL cache, the maximum CMR is for RC4 Algorithm and minimum is for Blowfish and RC5 Algorithm. Hence we conclude the CMR for Blowfish Algorithm is 0 percent for all type of caches. Thus in terms of CMR, the worst choice for I1 cache is skipjack Algorithm, for LLi cache is RC4 Algorithm, for D1 cache is RC4 Algorithm, for LLD cache is RC4 Algorithm, and for LL cache is RC4 Algorithm. In all cases where the CMR is a major constraint, the best choice considering all types of caches is Blowfish algorithm followed by RC5.

H. Read operations per second

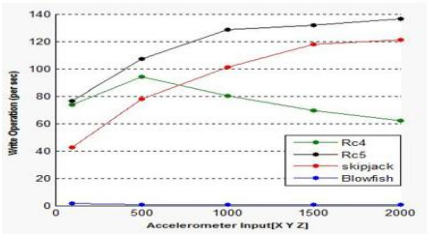
Considering the Read Operations per Second (ROS) vs. Accelerometer-Input, we find the maximum ROS to be for RC4 Algorithm and minimum for Blowfish Algorithm for any size of Accelerometer-Input. Also greater the number of Read operations per second easier it is for processor to Read the whole content in lesser time.



Graph 9: Read operations per second verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

In BSN where ROS is major issue, the best choice in terms of ROS is RC4 algorithm and worst choice is Blowfish algorithm for all sizes of Accelerometer-Input.

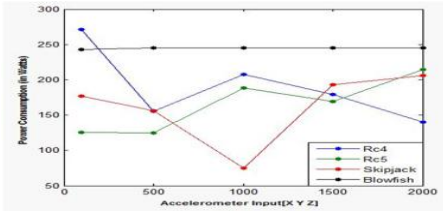
I. Write operations per second



Graph 10: Write operations per second verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

Focusing on the statistics for Write Operations per Second (WOS) vs. Accelerometer-Input, we find the maximum WOS to be for RC5 algorithm and minimum for Blowfish algorithm for any sizes of Accelerometer-Input. Also greater the number of write operations occurring per second, easier it is for processor to write the whole content in less time. In cases where WOS is major constraint, we conclude the best choice to be RC5 algorithm and worst to be Blowfish algorithm for all sizes of Accelerometer-Input.

J. Power consumption



Graph 11: Power consumption verses Accelerometer input for RC4, RC5, Skipjack and Blowfish

Focusing on the statistics obtained, we get the maximum power consumed to be by Blowfish Algorithm for any size of Accelerometer-Input. Also RC5 algorithm consumes minimum power for almost every Accelerometer-Input, except when the input is 1000 and 2000. This is because with increase in the size of input, the number of context switches per second decreases and the CPU load increases. For the input case of 1000, the CPU load is minimum and Skipjack Algorithm consumes minimum power. Further increasing the input, the CPU load increases as usual and thus consumes extra power. When the Accelerometer-Input is 2000, the power consumed by RC4 Algorithm, is minimum. Also an important factor to consider is that lesser the power, an Algorithm consumes, the more suitable it becomes for Small Board Computers (SBC). In BSN, where Power Consumption (PC) is a major constraint as BSNs are low power devices, the best choice as far as Power Consumption is concerned is RC5 algorithm for small size of input data and RC4 algorithm for large sizes of input data. The worst choice is Blowfish algorithm for all sizes of Accelerometer input.

V. ISSUES WITH THE ENCRYPTION ALGORITHMS

A. RC-4

It is considered to be efficient if key length is greater than 128 bits. Also the implementation of RC4 in WEP (Wired equivalent privacy), to secure wireless sensor networks, is not considered to be too efficient, the problem being the way in which it is used. RC4 also is not able to match the standards set by cryptographers for a secure cipher in various ways, and thus is not recommended for use in new applications [19]. The bytes RC4 produces are not always random- they contain small biases. [11] From the point of view of cryptography, this is not at all desired. Furthermore, it can happen that a double encryption of the message with the same key may output the plaintext again rather than cipher text because of the fact that the XOR function would result in the second operation reversing the first [11].

B. Blowfish

Blowfish is one of the fastest block ciphers; the problem comes when changing keys. Each new key requires a pre-processing to be done which is equivalent to encrypting about 4 kilobytes of text, this is comparatively very slow .This prevents its use in certain applications, but is not a problem in others, such as Splash ID. It also has a weakness in the decryption process over other algorithms in terms of the time consumption and serially in throughput [12].

C. RC-5

In the case of a sensor network, the costs of call setup and return outweigh the costs of the RC5 itself. RC5 is word-oriented. In comparison with RC4, RC5 consumes more code memory size Even though the RC5 algorithm can be small; the common RC5 libraries are too large to fit on a platform [1].

D. Skipjack

Its susceptibility to shortcut attacks, wherein the intruder can exploit some property of the encryption algorithm from which the key or plaintext can be determined in much less time than by exhaustive search, leads to a challenge in its use today [13]. The key length is longer making brute force very slow; however 80 bits can also be prone to brute force attacks [14].

VI. SITUATION BASED USE

A. RC-4

RC 4 is extremely efficient in software implementations. RC 4 can be considered secure if keys of length higher than 128 bits are used. If RC4 is used (i.e. any stream cipher technique) instead of DES (i.e. block cipher technique) in the output feedback mode, the net encryption and decryption time decreases at sensor node [19]. It is good for byte oriented operations [8]. RC4 is an extremely popular cipher for SSL/TLS connections. Also RC4 is very fast. Thus a fast encryption implies a less computation and therefore lower hardware requirements which are beneficial for service providers like Google [11].

B. Blowfish

Blowfish is considered to be suitable for wireless network application which exchange packets of comparatively small sizes [20]. Also Blowfish is free for use to anyone. This helps increase its usage in today's time [12]. Blowfish is said to be efficient in software, at least on some software platforms (it uses key-dependent lookup table; hence the performance depends on how the platform will handle memory and caches) [6].

C. RC-5

RC5 is patented by RSA and can be used as a replacement for DES with block size=64 bit and key size =56 bits [21]. It is used in IPSec Encapsulating Security Payload (ESP).It is used here in the CBC mode [15].It is even a good potential for use in Wireless Body sensor networks [2].It is used to provide security connections in LSWN (Low speed wireless networks) [25].

D. Skipjack

The algorithm was developed to be used in voice, fax and secure telephones e.g. the AT&T TSD-3600. It was also used in the first Fortezza Crypto Card. (By US Govt) [16].Skipjack is said to be immune to exhaustive search for time to come till the time we don't have further improvements in the exhaustive search techniques. The key of the Skipjack algorithm offers a greater margin of security than single encryption DES, but this margin is comparatively small and is overcome by the purely economic and technical considerations. Also the increasing key size in Skipjack does not necessarily increase costs as in single encryption DES [17]. It can also be used for data encryption in computer networks (Defense dept uses it in defense messaging system) [18].

VII. CONCLUSION

In table 1, the best and worst choice of algorithms has been depicted depending on the comparison factor considered. Small and large indicate the inputs of accelerometer (which algorithm is best and worst for small accelerometer reading and which for large). In the Cache miss rate, a distinction has been shown for various types of caches and the best and worst algorithm for cache miss rate of a particular type of cache. From Table 1, we inferred that the execution time for Blowfish algorithm is the maximum, while it is minimum for RC 4. Also the LOC and memory occupation of RC4 is minimum which is beneficial in case we have a memory limitation. For other cases when we consider the context switches per second or the page faults occurring per second, we get Blowfish as a better option than others. All the places where load on CPU is a constraint, CPU usage is best in case of RC4. For the cache miss rate, we consider all the types of caches namely I1, LLi, D1, LLD, LL cache. The cache miss rate is 0% in all cases for Blowfish and thus increases speed to a great deal. The corresponding Read and Write operations per second have RC4 and RC5 respectively as the best algorithms. The last factor of power consumption gives results that tell us that for small number of inputs RC5 is a good choice and as the number of inputs increase, the choice shifts from RC5 to RC4. Further the issue based evaluation and situational use of the algorithms has been depicted which will help decide the best suited algorithm for a given situation.

Table 1: A complete summation of all the results obtained.

Comparison Factors		Algorithms							
		Best Choice				Worst Choice			
		RC4	RC5	Skipjack	Blowfish	RC4	RC5	Skipjack	Blowfish
Execution time	Small	✔							✔
	Large	✔							✔
Memory Used		✔						✔	
LOC		✔						✔	
Context Switches per Second	Small				✔	✔			
	Large				✔	✔			
Page Faults per Second	Small				✔			✔	
	Large				✔	✔			
CPU usage	Small	✔							✔
	Large	✔							✔
Cpu time	Small	✔							✔
	Large	✔							✔
Cache Miss Rate	I1				✔			✔	
	LLi				✔	✔			
	D1				✔	✔			
	LLD				✔	✔			
	LL				✔	✔			
Read Operation per Second	Small	✔							✔
	Large	✔							✔
Write Operation per Second	Small		✔						✔
	Large		✔						✔
Power Consumption	Small		✔			✔			
	Large	✔							✔

REFERENCES

1. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar, "SPINS: Security Protocols for Sensor Networks", Mobile Computing and Networking, Rome, Italy, 2001.
2. Dhanashri H. Gawali and Vijay M. Wadhai, "rc5 algorithm: potential cipher solution for security in wireless body sensor networks (wbsn)", International Journal Of Advanced Smart Sensor Network Systems, July 2012.
3. Yao Minglin, TangShan Coll., Tangshan, Ma Junshuang, "Stream Ciphers on wireless sensor networks", Third International Conference on Measuring Technology and Mechatronics Automation, January 2011.
4. Shervin Amini, Richard Verhoeven, Johan Lukkien, Shudong Chen, "Toward a Security Model for a Body Sensor Platform", IEEE International Conference on Consumer Electronics, 2011.
5. Diaa Salama Abd Elminaam, Hatem Mohamed Abdual Kader, and Mohiy Mohamed Hadhoud, "Evaluating The Performance of Symmetric Encryption Algorithms", International Journal of Network Security, May 2010.
6. A website www.stackoverflow.com,(July 2, 2014).
7. Daojing He, Sammy Chan, Shaohua Tang, "A Novel and Lightweight System to Secure Wireless Medical Sensor Networks", Ieee journal of biomedical and health informatics, January 2014.
8. Xiaohua Luo, Kougen Zheng, Yunhe Pan, Zhaohui Wu, "Encryption algorithms comparisons for wireless networked sensors", IEEE lntemational Conference on Systems, Man and Cybernetics, 2004.
9. Daniel Tze Huei Lai, Rezaul Begg, Marimuthu Palaniswami, "Healthcare sensor networks-Challenges towards practical implementation", CRC Press Taylor &. Francis Group
10. Soham Banerjee, Divyashikha Sethia, Tanuj Mittal, Ujjwal Arora, Akash Chauhan, "Secure Sensor Node with Raspberry Pi", Impact 2013, 2013.
11. Mathew Greens, Attack of the week: RC4 is kind of broken in TLS, A Few thoughts on Cryptogaphic engineering: Attack of the week:RC4 is a kind of broken in TLS, (March 12, 2013), http://blog.cryptographyengineering.com/2013/03/attack-of-week-rc4-is-kind-of-broken-in.html (July 2, 2014)
12. A website http://en.wikipedia.org (July 2, 2014)
13. Skipjack Review, Skipjack Review, http://www.austinlinks.com/Crypto/skipjack-review.html, (July 3, 2014)
14. Skipjack, Skipjack-everything2.com, http://everything2.com/title/Skipjack, (April 29, 2014)
15. The esp cbc-mode cipher algorithms, draft-ietf-ipsec-ciph-cbc-01: The esp cbc-mode cipher algorithms, (July 2, 1997) http://tools.ietf.org/html/draft-ietf-ipsec-ciph-rc5-cbc-00, (July 2, 2014)
16. Skipjack, Skipjack, http://www.cryptomuseum.com/crypto/usa/skipjack.htm (July 2, 2014).
17. Kenneth W Dam, Herbert S Lin, "Cryptography's Role in Securing the Information Society", By Committee to Study National Cryptography Policy, Computer Science and Telecommunications Board, Division on Engineering and Physical Sciences, National Research Council,1996
18. US Congress, Office of Technology Assessment, "Issue Update on Information Security and Privacy in Network Environments", By DIANE Publishing Company, September 1995
19. Shish Ahmad, Mohd. Rizwan beg, Qamar Abbas, "Energy Efficient Sensor Network Security Using Stream Cipher Mode of Operation", Int'l Conf. on Computer & Communication Technology, 2010
20. Tingyuan Nie, Chuanwang Song, Xulong Zhi, "Performance Evaluation of DES and blowfish", Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on, 2010.
21. Introduction to cryptography, Introduction to cryptography, http://www.infosectoday.com/Articles/Intro_to_Cryptography/Introduction_Encryption_Algorithms.htm (June 28, 2014)
22. Coverity Scan: health-check, Coverity Scan-Static Analysis,(26 july 2013) https://scan.coverity.com/projects/661, (June 29,2014)
23. Coverity Scan: power-calibrate, Coverity Scan- Static Analysis, (26 july 2013), https://scan.coverity.com/projects/1732, (June 29, 2014)
24. ColinKing, ColinKing-Ubuntu Wiki, (4 June 2014) https://wiki.ubuntu.com/ColinKing/, (June 29, 2014)
25. Chou Fan, Jin Tan, Peng Zheng, "Low Speed Wireless networks research and simulation based on RC 5", Wireless communications, Networking and mobile computing,2009, Wicom '09. 5th international conference on, September 2009.