Efficient Utilization of Cryptographic Resources in Embedded Computing Systems

Muhammad Nabeel Asim[1], Muhammad Salman Khalid[1], Muhammad Idrees[2], Abdur Rehman[3]

Al-Khwarizmi Institute of Computer Science, University of Engineering and Technology, Lahore, Pakistan[1]
Department of Computer Science and Engineering, University of Engineering & Technology, Narowal, Pakistan[2]
Department of Computer Science and Engineering University of Gujrat, Gujrat Pakistan[3]

Abstract

The world is becoming a global village of intercon-nected electronic devices. With the cost of silicon going down, the number of connected devices is increasing day by day. With introduction of Internet of Things concept, embedded devices are finding their way into this global network. Eventually, more elaborate techniques would be required to maintain the security and privacy in this interconnected world. Application of Cryptographic techniques in embedded systems is now indispensable. This paper surveys the concerns and the constraints that need to be taken care of while selecting a particular cryptographic algorithm for embedded systems. We present an overview of cryptography and embedded systems. Then we provide some of the constraints in embedded systems development. In the end, we provide a survey of how different cryptographic techniques perform in an embedded environment.

Key words:
Cryptographic, embedded, computing system

1. Introduction

According to [2] almost 98% of all the 32-bit microprocessors deployed in electronic devices throughout the world are found in embedded systems. Since embedded systems have now become an integral part of the connected word, the information interchanged must be protected for privacy and security. The security breaches in embedded systems are equally disastrous as they are for traditional host PCs. They can not only compromise their own security but also the security of the other components of the system. It is a well-known concern that software defects are easily exploited by intruders for hacking into computer systems[3]. Software Applications that communicate over internet are most vulner-able to security risks [15]. A number of security risks that need to be taken care of are shown in figure1 (Also taken from [15]). Different types of security need to be ensured for different layers of the embedded systems architecture. The most common two concerns that are encountered in securing networked embedded systems are [16]:

1) Protecting data that is transferred from one place to another.

2) Protecting data within the embedded device.

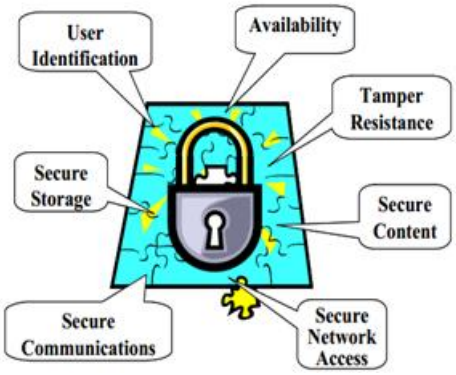

Fig. 1: Common Security Concerns Encountered in Embedded Systems

In this paper we mainly look at the ways to protect data that is transferred from one place to another, and how different implementations are suitable or well enough. The data in a public network passes through a number of points and links. For this data to be secured it must be communicated in ways such that it is of no use or value for intermediate nodes but end users who are authorized for this data can interpret this data. The techniques should be powerful enough that no intermediated node or eavesdropper can interpret it or draw meaningful data out of it and at the same time the computational complexities of such techniques should be such that embedded systems are able to handle them. This is because embedded systems are normally resource constrained from many aspects such as performance, memory and cost. Some of the most common techniques used for securing data are Cryptography[1], Digital Signatures and Digital Cer-tificates. Cryptography is a very important tool for securing network against malicious activities. It is the art and science of keeping the messages secures[6]. However their mathematical complexities often complicate the applications at hand. At the same time limitations enforced by embedded systems such as low memory[8], limited processing power further aggravate the complexities of the tasks. Therefore it is very important that optimized algorithms must be used and at the same time no compromise on security and privacy should be made. These

Manuscript received October 5, 2017
Manuscript revised October 20, 2017

limitations make the implementation of cryptographic algorithms in embedded systems a challenging task for the embedded developer[9]. In this paper, first we provide a general overview of cryptography and embedded systems. Then we outline embedded system constraints that may affect the performance of cryptographic algorithms. In the end we present a case study done by [7] to evaluate the performance of some well know cryptographic algorithm.

Fig. 2: Process of Cryptography

2. Literature review

A. cryptography

Cryptography came from two Greek words which mean secret writing [10]. Basically it is a technique of encrypting simple written information into some unreadable form so that others cannot understand it.

It is an art of hiding information in an abstract manner so that any irrelevant person cannot identify the content of the writing. The basic concept of cryptography is to share information among people in such an approach that other cannot understand it. It is a combination of words and mathematical techniques. This concept began in early days when humans started to communicate using written text. As time passes many cryptographic approaches have been developed which include shifting of alphabets and complex computer based encryption techniques. Basic concept of cryptography includes a message or information which is called plaintext or clear text and an encrypted mangled text called cipher text. Encryption is the process of converting plaintext into cipher text. Whole method is shown in figure2.

While new and modern cryptographic techniques are introduced, some people are always attempting to break these encryption methods. Both the groups try to keep themselves ahead of each other.

Cryptographic techniques include some algorithm for modifying the plaintext and a secret value which is known as key. Using the key reduces the effort to device new algorithms that can never be more secure and allow the decryption or reverse engineering of the encrypted text. Designing one secure algorithm and using different key combination can lead to a more secure cryptographic system. Having a strong cryptographic approach will allow everybody including bad guys to access the algorithm and the encrypted cipher text because without key the knowledge of algorithm is useless. The concept of key is similar to the simple combination lock system of daily use. It also has thousands of combinations

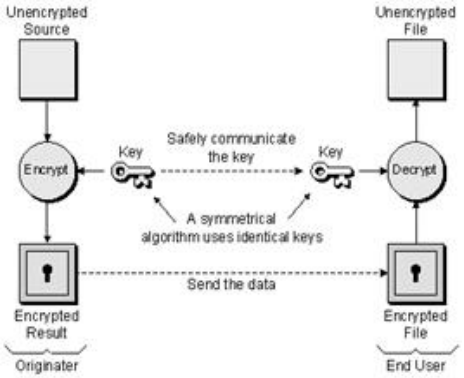

Fig. 3: : Private key cryptography scheme

Depending upon available digits. Nobody can open the lock even if he has access to it without having proper knowledge of the correct combination.

Cryptographic algorithms should be efficient so that the person having the key can decrypt the information within no time [11]. This is not an impossible task to break these algorithms without having the key. A person can break it by trying all the possible key combinations. So the best possible approach should take thousands or perhaps millions of year to break by trying all possible key combinations using powerful computers available. Let us consider an example of combinational lock having 4 numbers with each number ranging from 1 to 20. Let's assume that it takes about 10 seconds to try a combination for a person with the working key of combination. How much will it takes for a bad guy to try all possible combinations? There are 204 = 16000 combinations. If each takes 10 seconds then it would require about 18.5 days to break for worst case scenario (if you have to try all). For an average case it would require half of that time. To make it more difficult and time consuming the number can be increased to 99 from 20, then it would take more than 3 years to break that algorithm using all possible combinations. So by increasing key length the scheme is more secure. This is similar to the algorithms used in cryptography. Sometime they have variable length keys. While other times they have fixed length keys. There are two primary types of cryptography. 1) Private key cryptography 2) Public key cryptography

3. Types of Cryptography

A. Private key cryptography

In this type of cryptography, a message is encrypted using a secret but shared private key. This key is similar for both sender and receiver thats why it is also called a symmetric key cryptography. Encryption key is almost similar to the key used for decryption, but in some cases it can be partially

modified using a simple character transformation. The key remains a secret between communicating partners to share private information in public channels. Figure 3 refers to the symmetric key encryption scheme

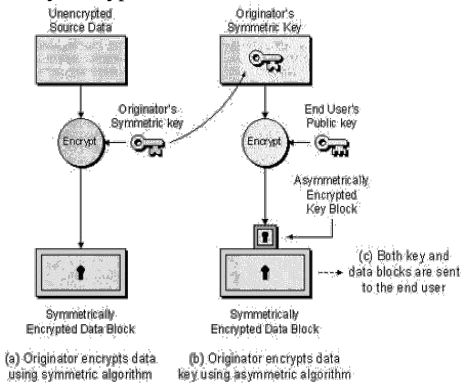

Fig. 4: Public key cryptography

Some examples of popular and well-respected symmetric algorithms include Two fish, Serpent, AES (Rijndael), Blowfish, CAST5, RC4, 3DES, and IDEA.

There are two basic fundamental methods of symmetric encryption techniques, Substitution and transposition. In sub-situation technique, letters are replaced by some other false characters, which actually make no sense and are chosen randomly or using specific pattern. For example, we replace letter d by z. In transposition technique, letters are just repo-sitioned to make the text incomprehensible. These algorithms are largely applied to large amount of data for the sack of fast operation.

In secret key algorithms, the important aspect of protecting data lies in sharing the key in a highly sophisticated and secure manner. All the strength of data privacy in the algorithm is based on the key. If the key is exposed to some unintended user or attacker, algorithm can do nothing to protect the data from being decrypted. Knowledge of algorithm is not sufficient to retrieve data because key is needed to decrypt the information, So when using the symmetric key techniques, the key sharing mechanism should be very secure.

B. Public key cryptography

Sharing the key in secure manner is an important issue, this is a significant disadvantage of symmetric key algorithms. So the solution to this problem is solved by introducing asymmetric keys that are used for the same purpose of encryption/decryption. This is also known as public key cryp-tography. Unlike secret key encryption, there are two keys associated with every individual, a public key that can be shared publically and a private key which is not exposed to anyone. Public key cryptography scheme is shown in figure 4. In this scheme, one can have same pair of keys for every cipher rather than to have a different secret key. Public key is used to encrypt the data ad it can be shared to everyone without any privacy, but for the decryption purpose a private key is needed. Public key cannot be used to decrypt data and it also do not provide any information about private key.

This type of cryptographic technique provide good support for user authentication and non-repudiation (Sender cannot deny that the data was encrypted using his private key). There is still a problem associated with this technique, someone hav-ing the public key of sender can easily decrypt the information encrypted by the private key of particular sender.

Authentication of the message is carried out by treating the message with private key. This results in generating digital signatures which can be verified by anybody by processing the signatures with public key of the signer and verifying the content of the message. This ensures that message was not modified and no one else has performed signature operation as the private key of the signer remains undisclosed.

C. Embedded Systems

Embedded Systems are computer systems that are a part of some larger system with often a dedicated and specific function to perform. Most often, the functions they perform are real-time and the computing resources are limited. These limited resources can include processing speed, power, mem-ory footprints etc. The embedded systems hardware usually consists of a microcontroller or a microprocessor placed on a PCB board with some peripherals on-chip and others off-chip. These peripherals include the following:

1- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485 etc.
2- Universal Serial Bus (USB)
3- Multi Media Cards (SD Cards, Compact Flash etc.)
4- Synchronous Serial Communication Interfaces: I2C, SPI etc.
5- Networks: Ethernet, Wi-Fi, etc.
6- Timers
7- Field buses: CAN-Bus, LIN-Bus, PROFIBUS, etc.
8- LCD Displays and Touch Panels
9- Analog to Digital/Digital to Analog Converters (ADC/DAC)
10- Discrete IO e.g. General Purpose Input/output (GPIO)
11- Analog to Digital/Digital to Analog Converters (ADC/DAC)
12- Debugging: JTAG, ISP, ICSP, BDM Port etc.

Processor architectures used are based on both Von Neu-mann and Harvard computer architectures. These microprocessors and microcontrollers are based on computer architectures such as ARM, ColdFire,

MicroBlaze, MIPS, NIOS II, PowerPC, SuperH, PIC and 8051.DSP processors are also used to perform calculation intensive tasks. With the advent of advanced fabrication techniques the trend is also shifting towards using multi core processors for improved performance and speed.

Embedded Systems are nothing without the software that operates them. The types of software that are used range from simple control loops and interrupt driven architectures to complex Real Time Operating Systems (RTOS). The type of software solution used depends upon the application. Simple software solutions like loops and interrupt driven code is used for small, specific and very simple applications such as displaying some data, controlling temperature, simple clock etc. Complex Operating Systems are used for more advance applications such as automotive, networking, telecommunica-tion where complex software stacks are required and a lot of tasks need to be taken care of. Some of the well-known RTOS include Mentor Graphics Nucleus RTOS, Wind Rivers VxWorks, Express Logics ThreadX and Microsoft Windows CE. Linux and its embedded variant are also used in embedded systems. Embedded Systems are involved in all applications from our daily lives. There applications range from digital watches and portable media players to advanced flight control systems for aircrafts. Be it military, consumer, cooking, auto-motive, space exploration or medical, we today owe our lives to embedded systems.

The world has now become a global village with hundreds of millions devices connected to the internet. It was inevitable that embedded computing systems have now eventually found their way into this global village. These embedded computing devices can be used to monitor environment, manage transport, monitoring health of people remotely, energy management like smart grid etc. This is the basic concept behind Internet of Things framework which is a network of uniquely identifiable embedded systems within the existing Internet infrastructure. In near future, it is estimated that the number of devices connected to the internet will surpass tens of billions.

With the exposure of embedded systems to the internet, the obvious security and privacy threats faced by ordinary host PCs are also faced by embedded systems[12]. Embedded communications also required to be secured using cryptogra-phy techniques [13]. The issue related to unauthorized data access, man in the middle attacks, Denial of Service attacks, user authentication are there and if care is not taken the consequences may be disastrous. For example consider a cars onboard computer that controls critical functions such as braking, lights and engines. If this system is also connected to the Internet and is no emphasis on security was given by the application developer, an attacker may be able to modify the functions of critical components and hence cause havoc for the driver. To solve security and privacy issues in embedded systems we must employ techniques that not only address the problem in question but also meet the constraints and limitations set by embedded devices. Embedded Systems also employ cryptography to solve such security and privacy issues. However the criterion for selection of cryptographic algorithm is now governed by the limitations set by embedded systems[14]. In this paper we explore the parameters of embedded systems and application in questions that help select the algorithm for cryptography. First we present the constraints and limitations of embedded systems that can affect the choice of algorithm. Then we present a survey of different experiments conducted to judge the feasibility of a cryptographic algorithm.

D. Embedded system constraints

Embedded systems are the most resource contained systems when it comes to meeting requirements of a specific task at hand. Almost, all the time, they contain just enough resources to meet the demands of the task at hand. Therefore, an efficient utilization of these resources must be accomplished [16]. In this section we present some of the design constraints posed by embedded systems to an application in question. These constraints [15] will help us understand the complexities of embedded systems while implementing cryptographic algorithms. These constraints are:

1- Power Requirement
2- Memory Footprint
3- Performance and Speed
4- Cost
5- Reliability

1) POWER REQUIREMENT: A considerable percentage of embedded devices are battery operated devices. A more power hungry system needs larger batteries to keep up with the power requirements. This adds to the weight and size of the device and ultimately to the cost of the system. If a processor inside an embedded system is busy executing some piece of code then it consumes more power than if it is sitting in idle condition[18]. So the algorithm designers must design algorithms that perform their task more quickly and efficiently.

Most of the embedded systems of today have special features for saving power such as Dynamic Voltage Scaling and Dynamic Frequency Scaling. With such features we can lower the Voltage and frequency on the fly. Both the voltage and frequency have direct impact on the power consumption. However, there are some tradeoffs that have to be made. By decreasing the frequency we not only lower the power consumption of the processor core but also its performance. Also if we decrease the operating voltage, we

also have to lower the frequency which also degrades performance. Since embedded systems usually have to meet tight timing constraints, this tradeoff between power and performance must be managed properly and efficiently by the system developer or the application programmer. Power consumption is the one of the constraints that we are going to explore and look how different algorithms affect the power consumption of an embedded device.

2) MEMORY FOOTPRINT: In an embedded system, especially a one based on microcontrollers, memory is a precious and a scarce resource. On microcontrollers memory is usually available in the form of on-chip ROM and RAM. The sizes of these on-chip memories are usually small. Although expanding memories in the form of adding external SDRAMs is an available option, this can only be done at the time of designing and manufacturing and adds to the overall cost of the system [18]. Expanding memory in embedded systems sometimes can cost more than the embedded system itself. Therefore it is up to the programmer to use the available memory as efficiently as possible. Minimizing code and data sizes are amongst one of the most painstaking job an embedded developer has to perform. Although todays compilers are intelligent enough and optimize code and data sizes very efficiently, an inefficient implementation of an algorithm eventually leads to expanded memory requirements. We also look at the memory consumption requirements of different algorithms.

3) PERFORMANCE AND SPEED: A real time embedded systems needs to maintain time critical deadlines. A variety of single core and multi-core devices are available today with tremendous processing speeds. However a faster processor means more power consumption. Therefore, the code written for embedded systems must be optimized for speed. Intelligent compilers today optimize code brilliantly but an algorithm is the major factor that determines the speed at which a particular task would be completed [17]. A slow algorithm will keep the processor busy and will definitely affect other real-time critical tasks in the system.

4) COST: Since embedded systems are usually a part of larger system it is favorable that their cost should be minimal as possible and at the same time they should perform their tasks flawlessly. There are many factors affecting the cost of embedded systems. Some of the most important factors are processor speed, memory, storage and development tools.

5) RELIABILITY: Most of the Embedded System devices run continuously for an extended period of time (not limited to months or years). It is highly demanded that they continue running without errors and faults. Even if errors or faults occur, embedded systems must employ methods and techniques to recover from these errors if the task at hand is a very critical one. Many times the embedded systems are employed in very extreme operating conditions. They may be exposed to high temperatures, shocks, moisture and vibrations. The hardware must be reliable enough to withstand all these extreme conditions. In environments like military and medical, military and medical grade components must be used which are much expensive as compared to consumer ones.

The reliability of embedded systems is not only limited to hardware. The software written for the embedded system must also be reliable enough to handle all the scenarios that it may encounter in the field. This software reliability must deal with security, real-time and safety aspects of the system. The failure to provide such reliability could have consequences ranging from minor nuisance to security breaches to even loss of lives. For example if a light is turned on or off with a little delay in a home smart grid environment then it may not be an annoyance but if there is a delay in application of brakes of an automobiles Anti-lock Braking System, lives could be at stake. Good programming practices, code reviews, code reuse, testing and quality assurance play an important role in making a particular software solution reliable.

4. Common encryption algorithms and their performance evaluation

In this section we present different common encryption algorithms used in practice. Then we look at their performance from different aspects such as power consumption, code size, data size and speed.

A. Encryption algorithms

1) DES- DATA ENCRYPTION STANDARD: The Data Encryption Algorithm is a symmetric key algorithm and a block cipher standardized by the National Bureau of Standards (NBS) in 1977. This algorithm uses a key of 64 bits. Out of these 64 bits eight bits are used for parity checking and 56 bits are the actual key used for encryption. The algorithm consists of 16 rounds. Feistel cipher serves as the basis for Data encryption standard. First an initial permutation IP is performed. Each block is broken into two parts. In each round a specially calculated round key is used and then permutation and substitution is performed. The details can be found in (Eckert 2006, p. 315). Finally, at the end a final permutation FP is performed which is the inverse of first initial permutation IP. For the source code implementation of this algorithm refers to (Schneier 2005). DES is now considered to be unsafe for almost all applications. A major reason for this is the short key length. Even a brute force approach can be easily used to break this algorithm.

2) DES TRIPLE DATA ENCRYPTION STANDARD: The major problem with DES was its short key length. Triple Data Encryption Standard extends the key length of the original Data Encryption Standard algorithm by applying Data Encryption Standard three times to each data block. This algorithm is considered to be practically safe although theoretical attacks have been demonstrated.

3) AES-ADVANCED ENCRYPTION STANDARD: The Advanced Encryption Standard is a symmetric key algorithm. It is based on Rijndael algorithm. It utilizes block and key length of variable sizes. The standard specifies key lengths of 128, 192 and 256 bits long. First the message is broken down into several smaller size blocks. First block is XORed with the first round key and is processed in 10 rounds using Substitution, Permutation, Diffusion and Key generation (Eckert 2006, p. 324). Decryption uses the same key-schedule for the round keys and this is performed using inverse of the encryption function. The decryption process is more complex so it takes more time. The AES is considered to be safe for many practical applications so far.

4) RC4 (RONS CODE 4): The RC4 algorithm was proposed by Ron Rivest in 1987. The secret key is composed of up to 2048 bits. This is large so breaking it is practically very difficult. The S-box is first initialized from 0 to 255 in a linear manner. Then a KBox is initialized with the secret and is repeated if necessary. After generation of the key stream, random bytes are created and XORed with the plaintext (Schneier 1996, 2005, p. 455). The large key size of 2048 bits makes this algorithm out of question for brute force attacks.

B. Performance evaluation

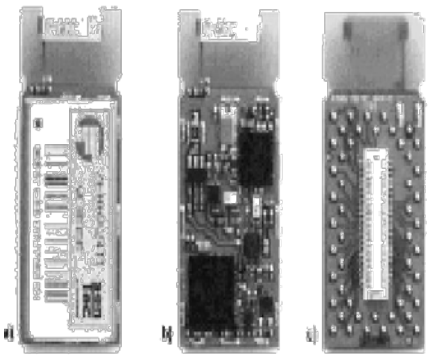

Fig. 5: ZEBRA Module Shielded, Unshielded and backside view

[6] has conducted the performance evaluation of some of the most well know encryption algorithms. The performance parameters evaluated for any encryption algorithm are speed, power consumption and memory requirement. The author has used a point based scheme to judge the feasibility of a particular algorithm and also to develop a comparison between them. For evaluation purposes the author has used a ZigBee Enabled Board Application for Radio Applications (ZEBRA) from senTec Elektronik GmbH. It is shown in figure 5 (Also taken from [6]). The ZEBRA module has an

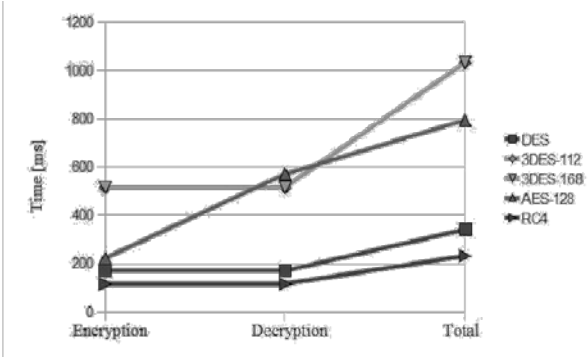

Fig. 6: Comparison of times taken for both Encryption and Decryption

HCS8GT60 microcontroller manufactured by Freescale Semiconductors. Now we will present the results calculated by the author for different encryption algorithms. The algorithms compared were Data Encryption Standard and its two variants, Advanced Encryption Standard and Rons Code 4.

1) DES: A 1000 encryption and decryptions were performed. The duration for 1000 encryptions was calculated to be 342.016 ms. Total energy consumption came out to be 375.0793 uWh. Disk space utilized was 16110 bytes. On the authors scale this algorithm scored 68 points

2) 3DES-112: A 1000 encryption and decryptions were performed. The duration for 1000 encryptions was calculated to be 1033.202 ms. Total energy consumption came out to be 1144.3418 uWh. Disk space utilized was 16133 bytes. On the authors scale this algorithm scored 25 points.

3) 3DES-168: A 1000 encryption and decryptions were performed. The duration for 1000 encryptions was calculated to be 1033.168 ms. Total energy consumption came out to be 1157.1705 uWh. Disk space utilized was 16174 bytes. On the authors scale this algorithm scored 24 points.

4) AES: A 1000 encryption and decryptions were performed. The duration for 1000 encryptions was calculated to be 795.072 ms. Total energy consumption came out to be 886.7849 uWh. Disk space utilized was 23384 bytes. On the authors scale this algorithm scored 34 points.

5) RC4: A 1000 encryption and decryptions were performed. The duration for 1000 encryptions was calculated to be 232.736 ms. Total energy consumption came out to be 284.9429 uWh. Disk space utilized was 2374s bytes. On the authors scale this algorithm scored 84 points. Now to show a comparison of the techniques used above we present a

number of Figures from [6]. The Figures 6 , 7 and 8 compare the Duration, Energy Consumption and Memory Requirements of all the algorithms presented above respectively and are self-

6) Explanatory. The Figure 9 gives the final verdict by showing the overall performance of all algorithms using the point based scheme. It shows that RC4 algorithm is the most suited one if all the embedded systems constraints are to be taken into account for this ZEBRA module.

5. Conclusion

The key motivation for writing this paper is to have a glimpse of the performance of various cryptographic algorithms in an embedded systems environment. Embedded Systems is an emerging field but even then it is expected that the numbers of embedded device connected to the internet would surpass tens of billions in the near future. Therefore in this connected environment issues of privacy and security will be huge. Cryptography is a very active research area not only because of the security concerns but also because of its mathematical importance. This research focuses on limitations posed

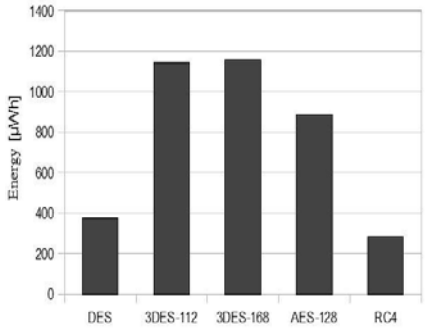

Fig. 7: Energy Consumption.

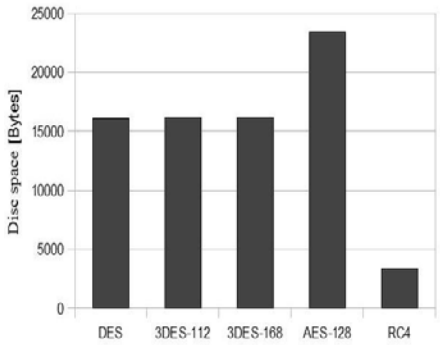

Fig. 8: Memory Requirements

by embedded systems on cryptographic implementations and a how different algorithms perform under these limitations.

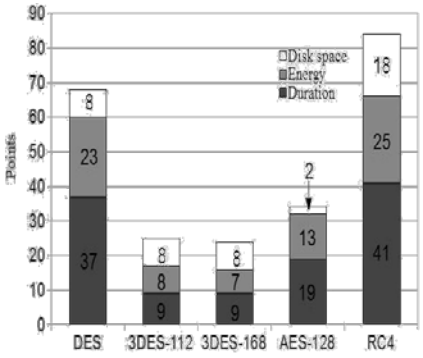

Fig. 9 Evaluation on point based scheme

REFERENCES

[1] Diffie, Whitfield, and Martin Hellman. ”New directions in cryptography.” IEEE transactions on Information Theory 22.6 (1976): 644-654.

[2] Wolf, Wayne, and Jan Madsen. ”Embedded systems education for the future.” Proceedings of the IEEE 88.1 (2000): 23-30.

[3] McGraw, Gary, and Greg Hoglund. ”Exploiting software: How to break code.” Invited Talk, Usenix Security Symposium, San Diego. 2004.

[4] Kocher, Paul, et al. ”Security as a new dimension in embedded system design.” Proceedings of the 41st annual Design Automation Conference. ACM, 2004.

[5] Anoop, M. S. ”Security needs in embedded systems.” IACR Cryptology ePrint Archive 2008 (2008): 198

[6] Schneier, B. ”Applied Cryptography, ohn Wiley & Sons Inc.” New York, New York, Sa, 4 (1996).

[7] Alexander, Miller, and Ing Gunar Schorcht. ”Embedded Systems Secu-rity: Performance Investigation of Various Cryptographic Techniques in Embedded Systems.” Proceedings of IT Security Conference for the Next Generation. 2011.

[8] Sharif Mansouri, Shohreh. Design and Implementation of Efficient and Secure Lightweight Cryptosystems. Diss. KTH Royal Institute of Tech-nology, 2014.

[9] Paar, Christof, and Jan Pelzl. Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media, 2009.

[10] Stinson, Douglas R. Cryptography: theory and practice. CRC press, 2005.

[11] Poschmann, Axel York. ”Lightweight cryptography: cryptographic en-gineering for a pervasive world.” PH. D. THESIS. 2009.

[12] Kermani, Mehran Mozaffari, et al. ”Emerging frontiers in embedded security.” VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), 2013 26th International Conference on. IEEE, 2013.

[13] Dhillon, Parwinder Kaur, and Sheetal Kalra. ”Elliptic curve cryptogra-phy for real time embedded systems in IoT networks.” Wireless Networks and Embedded Systems (WECON), 2016 5th International Conference on. IEEE, 2016.

[14] Fysarakis, Konstantinos, et al. ”Embedded Systems Security Chal-lenges.” PECCS. 2014.

[15] Kocher, Paul, et al. ”Security as a new dimension in embedded system design.” Proceedings of the 41st annual Design Automation Conference. ACM, 2004.
[16] Anoop, M. S. ”Security needs in embedded systems.” IACR Cryptology ePrint Archive 2008 (2008): 198.
[17] Duranton, Marc. ”The challenges for high performance embedded sys-tems.” Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on. IEEE, 2006.
[18] Oliveira, Lizandro, Jlio CB Mattos, and Lisane Brisolara. ”Survey of memory optimization techniques for embedded systems.” Computing Systems Engineering (SBESC), 2013 III Brazilian Symposium on.

Muhammad Nabeel Asim received his Bachelor degree from University of Management and Technology (UMT) and Master’s degree in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan. Currently working as Research Officer at Al-Khawarizmi Institute of Computer Science (KICS) University of Engineering and Technology (UET), Lahore Pakistan. His research interests are Bioinformatics, Artificial Intelligence, Machine Learning, Network Security, and Embedded Systems.

Muhammad Salman Khalid completed his Bachelor degree from University of Management and Technology (UMT) in Electrical Engineering and Masters in Computer Engineering from University of Engineering and Technology, Lahore, Pakistan. He worked as Lab engineer in University of Management and Technology. He also have worked in Network Architecture Lab (NAL) at EPFL, Switzerland.

Muhammad Idrees Completed his PhD from UET Lahore in computer science. He worked as an Assistant Professor and Head of Department of Computer Science and Information Technology, in BZU, Lahore and UoS, Lahore Campus for four years since 2011 to 2016. Now, he is working as an Assistant Professor and Head of Department in Computer science and Engineering, in University of Engineering and Technology Lahore, Narowal Campus since 2016 to date. His research interests include Bioinformatics, Databases, Software Engineering, Network Security, Artificial Intelligence and Embedded Systems.

Abdur Rehman Completed his PhD from UET Lahore in computer science. During his PhD, he has been part of Al-Khawarizmi Institute of Computer Science, UET Lahore for 8 year working as a researcher on different posts. His core expertise are in the field of machine learning and his focused area of research is “Text Classification”. He is currently working as Assistant Professor in Department of Computer Science, University of Gujrat.