Indian Journal of Science and Technology, Vol 9(29), DOI: 10.17485/ijst/2016/v9i29/90026, August 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Optimized Hardware Crypto Engines for XTEA and SHA-512 for Wireless Sensor Nodes

Ahmed Al Maashri*, Lavanya Pathuri, Medhat Awadalla, Afaq Ahmad and Mohamed Ould-Khaoua

Department of Electrical and Computer Engineering, Sultan Qaboos University, Muscat, Sultanate of Oman; amaashari@squ.edu.om, m104214@student.squ.edu.om, medhatha@squ.edu.om, afaq@squ.edu.om, mok@squ.edu.om

Abstract

Objectives: This study proposes an optimized power-efficient cryptosystem that is suitable for Wireless Sensor Networks. **Methods:** A number of cryptographic algorithms have been proposed to secure Wireless Sensor Networks. However, these compute-intensive and power-hungry algorithms do not take into consideration the limitations of resource found on the sensor nodes. We propose profiling some of these popular algorithms to identify the speed bottlenecks and develop hardware accelerators that maintain the real-time performance, with an efficient use of power. **Findings:** The proposed optimizations to the hardware accelerators were mapped to reconfigurable computing devices. Results show that the performance of the proposed hardware outperforms the software implementation running on contemporary CPU by up to 21.9×. In addition, the results indicate that the hardware is efficiently managing its power budget. **Application:** These accelerators can be utilized in heterogeneous system architectures, where the CPU controls the overall operations, and the accelerators efficiently perform the necessary encryption and decryption.

Keywords: Cryptography, FPGA, Power Efficiency, Reconfigurable Computing, WSN

1. Introduction

Wireless Sensor Networks (WSNs) consist of a complex linkage of self-determining sensor nodes that sense, measure, and gather information from a remote environment. They send information via means of wireless communications. WSN sensor nodes are commonly deployed in the field of observation in a manner that facilitates a smooth and uninterrupted inter-nodal and gateway communication. A sensor node varies in dimensions from that of a shoebox down to the size of a grain of dust[1]. Nowadays, there is a drastic reduction in the size of the sensor nodes with equal or even greater sensing capability.

WSNs can be utilized in several application domains. For instance, they can be deployed in harsh environments (e.g. battlefields and nuclear plants), short-range environments (e.g. temperature and smoke detection), and even large area deployment (e.g. forests, agricultural land, and buildings). Similarly, WSNs can be used for intrusion detection, surveillance, natural disaster relief, seismic survey, industrial, and underwater and Oceanography[1].

WSNs are versatile in the way they are deployed, as they can be deployed autonomously in single-sink, multi-sink networks, single hop, multi-hop networks, flat, and hierarchical network topologies[2].

Despite the several advantages of using WSNs, however, these networks suffer from a number of drawbacks. For instance, sensor nodes are running off a tiny battery, have limited hardware resources, and transmit information in short communication range with low bandwidth[3]. In addition, WSNs usually employ broadcast for transmission. These limitations make nodes in WSNs more vulnerable to security attacks.

It is essential to protect the information that is exchanged in a WSN from malicious attacks. However, cryptographic algorithms are usually compute-intensive, which results in an inefficient execution over the nodes' processors. This inefficiency is mainly observed in slow execution time and higher power consumption.

*Author for correspondence

One approach to overcome this issue, is to develop a hardware co-processor that is customized for such application domains. In such case, the processor can offload the cryptographic workloads to the co-processors, which can execute these workloads in real-time and at lower power budget. The use of customized co-processor comes with a number of advantages such as independence from the Operating System, higher performance, lower power consumption, and cost effectiveness for medium to large applications[4].

This paper makes two main contributions. First, the paper profiles two cryptographic algorithms to better understand their computational structure. Then, the paper proposes modifications to the hardware architecture in order to improve the performance, while increasing the power efficiency.

The rest of the paper is organized as follows. Section 2 gives an overview of security in WSNs, while Section 3 presents the profiling of the algorithms. This is followed by Section 4, which describes the proposed modifications made to the hardware architecture of the accelerators. Section 5 discusses the process of porting the hardware accelerators to the platform FPGA, which is followed by the performance results in Section 6. Finally, Section 7 concludes the paper.

2. Security in WSN

Cryptographic algorithms, initially developed to secure traditional computing systems, have proved themselves to be slow and power hungry in WSNs. To support this claim, we present Figure 1. This figure shows the execution time for encrypting a fixed chunk of data (100KB) by several well-known algorithms[5,6]. It is clear from the figure that even when these algorithms are executed on a contemporary CPU, the execution time is relatively slow. Therefore, mapping these algorithms to a sensor node would result in an even slower execution time.

In a previous survey study that we have conducted[7], we have reported the cryptographic algorithms that are utilized in WSNs. As shown in Figure 2, symmetric keys were used in 33% of the works that were covered in the study, while 11% have utilized elliptic curve cryptography. Similarly, Block ciphers were used in 7% of the surveyed works, while hash functions and AES have scored 5% each. Unlike other algorithms, AES is not popular in WSN. This could be explained as follows. Given the level of computational complexity of the algorithm, the execution time will be prohibitive considering the limited resources on the sensor nodes. This might lead developers to trade off level of security with speed. Such compromises are always possible, especially in certain types of applications.

This has motivated the community to developed customized hardware engines that are capable of running these algorithms at higher speeds. These engines could be mapped to either an Application-Specific Integrated Circuit (ASIC) or a Field-Programmable Gate Array (FPGA). The latter is a more favorable alternative since FPGAs are reconfigurable computing devices that can speed up algorithms, while operating at a low power budget. Also, FPGA devices come in smaller footprints and lighter weight – making them a fit for sensor node applications. The same study presented above[7], has surveyed previous efforts in mapping cryptographic algorithms to FPGA hardware for WSNs, as illustrated in Figure 3. The

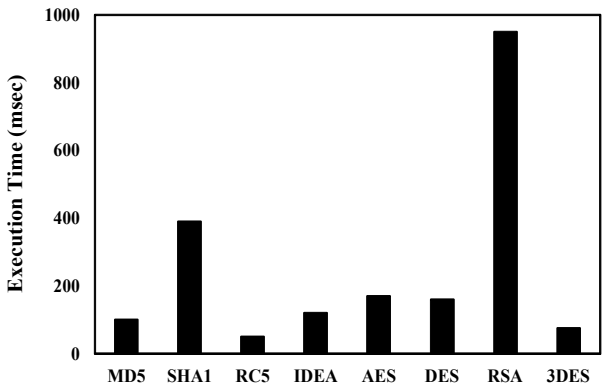

Figure 1. The execution time for a number cryptographic algorithms to encrypt 100k bytes of data.

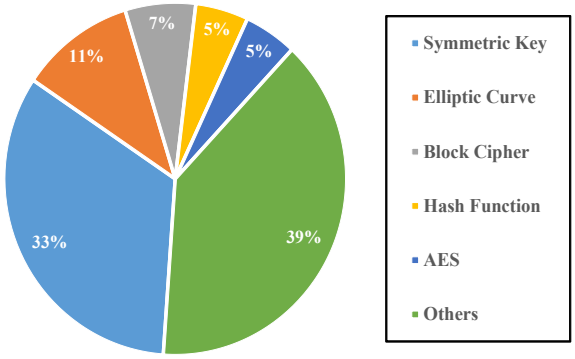

Figure 2. Cryptographic algorithms that are used in WSN[7]. This chart gives an indication of the popularity of these algorithms.

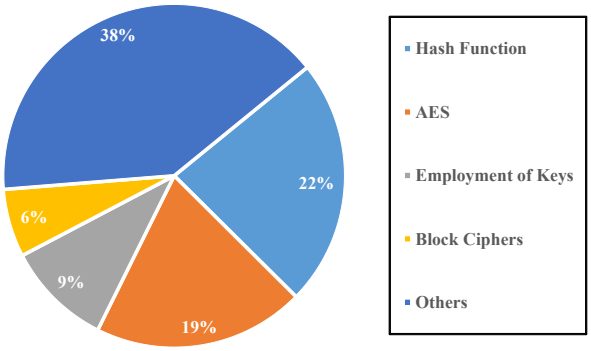

Figure 3. Cryptographic algorithms that are were mapped to FPGAs to be used in WSNs[7].

figure shows that extensive efforts have been invested in developing FPGA accelerators for hash functions (22%) and AES (19%). Others' proposals varied between algorithms that employ a key (9%) or those that utilize block cipher techniques (6%). The algorithm Extended Tiny Encryption Algorithm (XTEA) is an example of a block cipher algorithm.

Based on what was presented above, we are motivated to explore the optimization of hardware accelerators such that they perform well, while maintaining a reasonable power profile. For this purpose, we have selected two commonly used algorithms; namely, XTEA and SHA-512. The reason for choosing these two algorithms is that they are gaining popularity in WSNs. Additionally, these algorithms do not possess the same computational structure, something that requires care while designing the hardware architecture of the accelerators.

3. Profiling of Algorithms

Security algorithms are composed of computational structures, which can be complex. The complexity of these structures determine the execution time of these algorithms. Therefore, it is essential to understand these structures through software profiling. Based on the outcomes of the profiling, one can identify the bottlenecks that require acceleration, since most of the efforts need to be invested in optimizing those specific bottlenecks in order to speed up the execution time. This section discusses in details the software profiling that was carried out to identify the bottlenecks in the two algorithms; namely, XTEA and SHA-512.

Both XTEA and SHA-512 were profiled on a 3.4GHz Intel Core i7 platform, with 4GB DDR3 RAM. The platform is running a 64-bit Linux operating system. The source code of the algorithms was obtained from[8]. Note that the results reported in the following subsections are the average of 15 runs for each configuration in each algorithm.

3.1 XTEA Profiling

XTEA was profiled using different configurations by varying the block size of the data and the size of the key. Two block sizes are used; namely, 8 and 16 bytes. Those blocks sizes are tested against key sizes that range from 8 to 128 bytes. Figure 4 illustrates the execution time of XTEA for each configuration. Note that smaller blocks of data incur more processing overhead, which results in slower execution time as observed in the figure.

Extensive tests reveal that XOR operations dominate the execution time of the XTEA algorithm. These operations can be parallelized to improve the overall performance.

3.2 SHA-512 Profiling

Similar to XTEA, SHA-512 was profiled using the same configurations (i.e. block sizes of 8 and 16 bytes with a key size that ranges from 8 to 128 bytes). **Figure 5** shows the execution time for each configuration.

Unlike XTEA, results show that the execution time is dominated by memory transactions, which drastically impact the performance. Therefore, customized cache structures and data reuse techniques could potentially improve the performance of the algorithm.

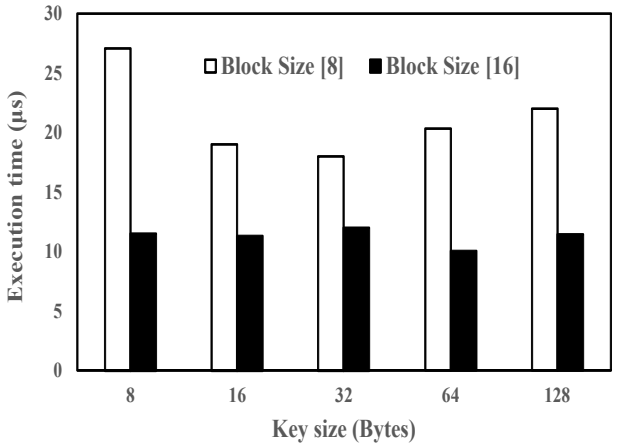

Figure 4. The execution time of XTEA algorithm using different configurations by varying block and key Sizes, while using the same message block.

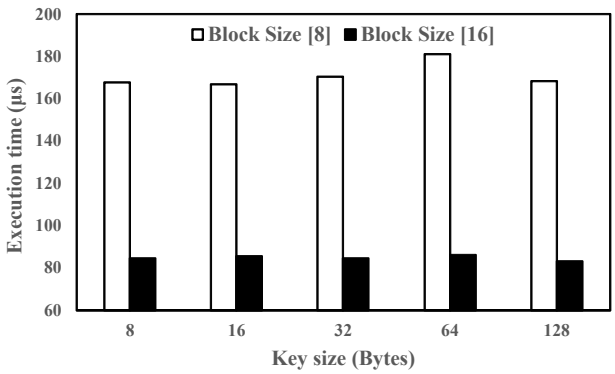

Figure 5. The execution time of SHA512 algorithm using different configurations by varying block and key Sizes, while using the same message block.

4. Hardware Architecture of the Accelerators

Developing a hardware architecture for an accelerator requires a comprehensive understanding of the structure of the algorithm. Most importantly, the designer must optimize the accelerator to improve the performance. In our case, the design must also be optimized for low power consumption, since the accelerator will be ported to a sensor node that is running off a battery. Therefore, the design need to be implemented with the smallest possible area, which will result in lower power consumption. However, care should be taken such that performance is not drastically compromised. There are a number of researchers who have investigated prime factors for the use of cryptographic systems, energy efficient cryptographic keys and mapping cryptographic algorithms to hardware, some of such works can be found in research articles[9-12].

The following subsections discuss the details of the hardware architecture for the accelerators. These architectures are modified and optimized variations of the design developed by[13].

4.1 XTEA Architecture

Figure 6 shows a block diagram of the XTEA encryption engine. The accelerator accepts three inputs: the *key*, the *message* to be encrypted, and *load*. On the other hand, the accelerator has two output signals: *enc_msg* and *done*. The *key* and the *message* are loaded when the signal *load* is asserted for one cycle. Once the encryption of the message is completed, the *done* signal will be asserted for one cycle, and the encrypted message can be sampled at the signal *enc_msg*.

Similarly, Figure 7 depicts the XTEA decryption accelerator. The same inputs are used here, except that the *message* signal is replaced with *enc_msg*.

4.2 SHA-512 Architecture

Figure 8 shows a block diagram of the signaling and components of the SHA-512 accelerator. There are three inputs; namely, *cmd* (the command), *cmd_en* (command enable) and *message*. The accelerator calculates the message digest, which can be sampled at *hash_digest* once the signal *done* is asserted.

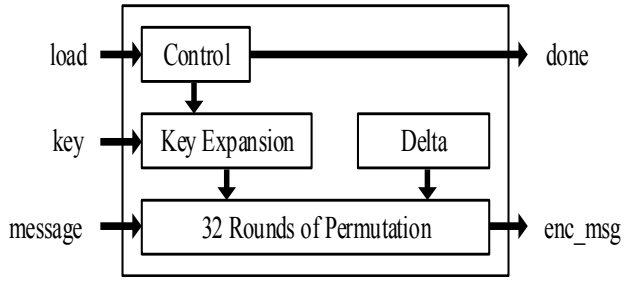

Figure 6. A block diagram showing the XTEA encryption accelerator.

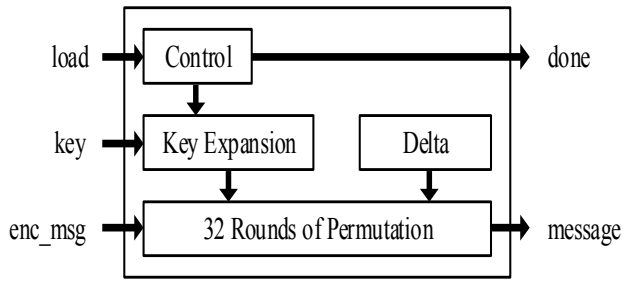

Figure 7. A block diagram showing the XTEA decryption accelerator.

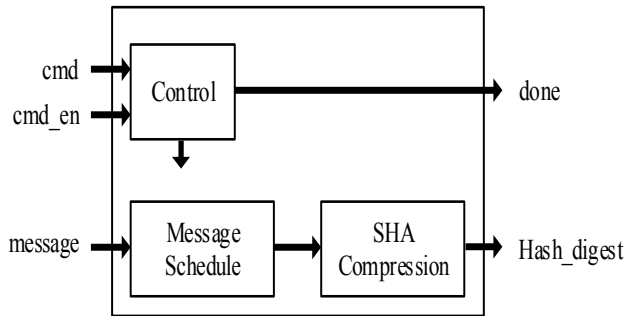

Figure 8. A block diagram showing the SHA-512 accelerator. The internal bus has a bit width of 32.

5. Prototyping the Accelerators to FPGA

There are different approaches for implementing the architectures discussed above. These are: iterative, combinatorial, and pipelining. Each of these approaches has its own pros and cons when it comes to performance and area when mapped to hardware logic.

To put things into perspective, consider the implementation of the crypto accelerators. In an iterative architecture, the input message is encrypted or decrypted iteratively using the same hardware resource. This way, the consumed area is reduced to decrease the overall power consumption. On the other hand, pipelining introduces additional stages of processing. The pipeline allows overlap in time, hence improving the overall throughput. Figure 9 shows how each one of these approaches perform, when mapped to FPGAs, while implementing the same design.

It is evident from the figure that the iterative implementation yields a good trade-off between throughput and resource utilization. This makes the iterative approach ideal for wireless sensor nodes. The reader is reminded that there is a direct relationship between the area and total power consumption. The more area is utilized, the higher switching activity, which leads to higher power consumption.

Therefore, in this paper we follow the iterative approach in implementing all the accelerators. The implementation was developed in Verilog HDL, with internal bus size of 32 bits.

The accelerators were validated using ModelSim Simulator[16]. The simulation environment was injected with different test cases to cover all corner scenarios. The simulation was followed by synthesizing the HDL code. We use Quartus II design software, from Altera[17], to synthesize the accelerators and map them to the FPGA. All the accelerators were mapped to the Altera DE0 platform FPGA, from teras IC[18].

In order to validate the accelerators on the FPGA, we have developed a feeder logic that utilizes UART serial interface. The feeder supplies the plain text from a PC to the FPGA. The platform FPGA was operated at 50MHz. The plain text input was readily available as soon as the accelerators are ready to accept a new input.

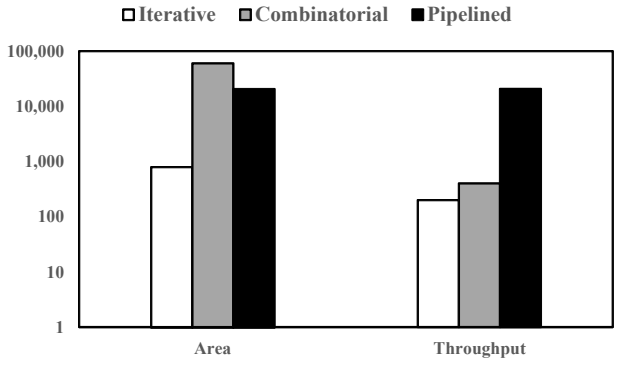

Figure 9. Throughput and slices for four architectural types in FPGA[17,18]. The area is this figure is based on the FPGA slice utilization.

6. Experimental Results

After validating the functionality of the XTEA and SHA cores, we have conducted a number of experiments to measure the performance of the accelerators. In all of these experiments, we use the same message size of 1.6 KB.

Figure 10 shows the speedup gain when mapping the accelerators to FPGA, compared to the CPU platform that was described earlier. For instance, the XTEA accelerator has speeded up the execution time by 16.4×, while SHA-512 accelerator has scored a speedup gain of 21.9×. The reason for such significant gain is the use of the parallel resources available on the FPGA. In addition, data reuse techniques and customized cache architectures have significantly improved the memory access, hence boosting

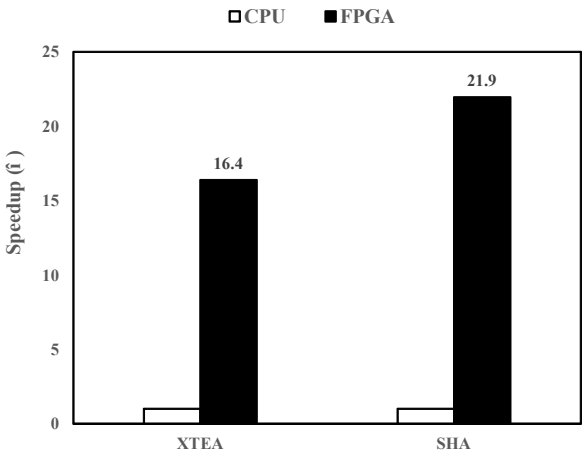

Figure 10. Speedup gained by prototyping the accelerators to FPGA normalized to the CPU platform. The experiments were conducted using a block size and key size of 16 bytes each.

the overall performance. These caches were constructed using Block-RAMs, which are hardware structures available on the FPGA.

Power efficiency is another important metric. In this context, power efficiency represents the amount of work done per unit of energy/power[19]. This is a useful metric, as it gives an indication of how efficient is the architecture in utilizing the supplied power to perform a given task. Therefore, a higher power efficiency signifies a more efficient architecture. Power efficiency can be computed as follows:

$$Power\ Efficiency = \frac{Performance}{Power} \quad (1)$$

The FPGA platform that was used to map the accelerator is running at 5 Watts, while performance is computed as the number of messages that can be processed by the accelerator per second. In other words:

$$Performance = \left(\frac{1}{Execution\ Time} \right) / 1000 \quad (2)$$

Here, the performance is measured for 1000 messages, as this will yield more meaningful numbers.

Figure 11 illustrates the power efficiency of the two accelerators that were mapped to the FPGA. The figure shows that the power efficiency of the XTEA and SHA accelerators have scored 290 and 51, respectively. Such higher power efficiency is possible since the FPGA is running at low frequency (50 MHz), while maintaining the throughput necessary to process the tasks in real time.

We compare our improved version of the XTEA and SHA-512 accelerators with other works. Table 1 shows a comparison in terms of performance and area between our design and previous works. In the table, throughput is used as a measurement of performance. Likewise, the utilization of FPGAs' Lookup tables (LUTs) is used to estimate the area utilization. It is clear from the table that our optimized accelerators make good tradeoff between performance and area utilization.

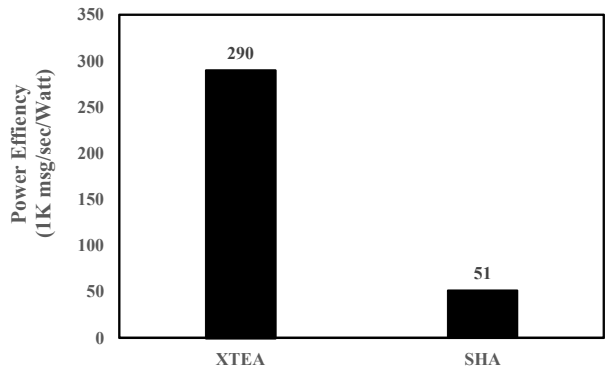

Figure 11. Power efficiency of the accelerators.

Table 1. A Performance-Area comparison with previous works

XTEA				
Design	**Target Device**	**Frequency (MHz)**	**Throughput (Mbps)**	**Area (LUTs)**
(20)	Cyclone III	60	100	6,590
(21)	Virtex 5	200	18,286	17,310
Our Design	Cyclone IV	50	1,440	1,678
SHA-512				
Design	**Target Device**	**Frequency (MHz)**	**Throughput (Mbps)**	**Area (LUTs)**
(22)	Virtex 5	64.45	117.8	556
(23)	Virtex 4	33.147	1,616	10,328
Our Design	Cyclone IV	50	1,444	5,660

7. Conclusions

The work presented in this paper attempts to optimize cryptography hardware accelerators that can be utilized in WSNs. The paper profiled two commonly used algorithms; namely, XTEA and SHA-512. Then, based on the profiling result, the computational structure of the algorithms was analyzed to identify the performance bottlenecks. The accelerators were implemented using the iterative approach and were validated both in simulation and on platform FPGA. Results show that the XTEA algorithm can be speeded up by 16.4×, while SHA-512 is speeded up by 21.9×. On the other hand, the power efficiency of the cryptography algorithms can be further improved by 290 for XTEA and 51 for SHA-512.

For future work, the authors are currently working on porting other cryptography algorithms to FPGAs.

8. Acknowledgment

This work was supported by a grant from the Sultan Qaboos University (IG/ENG/ECED/15/02). The authors would like to thank Mr. Sulaiman Al Habsi for his help in validating the hardware cores on the platform FPGA.

9. References

1. Hodjat A, Verbauwhede I. A 21.54 Gbits/s fully pipelined AES processor on FPGA. Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. 1025852: IEEE Computer Society. 2004; 308–9.
2. Huang C-W, Chang C-J, Lin M-Y, Tai H-Y. The FPGA implementation of 128-bits AES algorithm based on four 32-bits Parallel operation. Proceedings of the: The First International Symposium on Data, Privacy, and E-Commerce. 1338092: IEEE Computer Society. 2007; 462–4.
3. Yee Wei L, Havinga PJM. How to secure a wireless sensor network? Proceedings of Second International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Melbourne, Australia, IEEE. 2005. p. 89–95. ISBN 0-7803-9399-6.
4. Nadeem A, Javed MY. A Performance comparison of data encryption algorithms. Proceedings of First International Conference on Information and Communication Technologies (ICICT 2005). 2005. p. 84–9.
5. Avancha S, Undercoffer J, Joshi A, Pinkston J. Security for wireless sensor networks. Wireless sensor networks: Kluwer Academic Publishers. 2004; 253–75.
6. Stelte B. Toward development of high secure sensor network nodes using an FPGA-based architecture. Proceedings of the 6th International Conference on Wireless Communications and Mobile Computing, Caen, France. 1815521: ACM. 2010. p. 539–43.
7. Pathuri L, Al Maashri A, Ahmad A, Khaoua MO, Awadalla M. Securing wireless sensor networks using customized hardware crypto engine. Proceedings of National Symposium on Innovations in Information Technology (NSIIT'15). 2015; 54–7.
8. Limited A. mbed TLS Source Code: Fully open-source. 2015. Available from: https://tls.mbed.org/source-code.
9. Ahmad A, Al Busaidi SS, Al Maashri A, Awadalla M, Rizvi MAK, Mohanan N. Computing and Listing of Number of Possible m-Sequence Generators of Order n. Indian Journal of Science and Technology. 2013; 6(10):5359–69.
10. Ahmad A, Samarth A, Al Maashri A, Al Busaidi SS, Al Shidhani A. On determination of LFSR structures to assure more reliable and secure designs of cryptographic systems. Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2014, At Amity University, Noida, India, IEEE. 2015; 1:163–7. ISBN: 978-93-83083-99-2. Doi: 10.1109/ICRITO.2014.7014681, 1-5.
11. Ahmad A, Al Busaidi SS, Al Lawati A, Tarhuni N. On energy efficient cryptographic keys. Proceedings of National Symposium on Innovations in Information Technology (NSIIT'15). 2015; 92–4.
12. Thasneem PT, Salim PT, Vigneswaran T. FPGA implementation of hiding information using cryptography. Indian Journal of Science and Technology. 2015; 8(19):110–6.
13. Barari A. Digital hardware IP cores XFA. 2015. Available from: http://www.amirbarari.ir/xfa/hardwareipcores.html
14. Krukowski L, Sugier J. Organization of AES Cryptographic Unit for Low Cost FPGA Implementation. Proceedings of Third International Conference on Dependability of Computer Systems, DepCos-RELCOMEX '08, IEEE 10.1109/DepCoS-RELCOMEX. 2008; 36:347–54.
15. Kretzschmar U, Astarloa A, Jesus L Z, Bidarte U, Jimenez J. Robustness analysis of different AES implementations on SRAM based FPGAs. Proceedings of Third International Conference on Reconfigurable Computing and FPGAs (ReConFig 2011), IEEE. 10.1109/ReConFig. 2011; 80:255–60.
16. Mentorgraphics/Altera. Model Sim-Altera Software. 2015. Available from: https://www.altera.com/products/design-software/model---simulation/modelsim-altera-software.html
17. Altera. Quartus Prime Software. 2015. Available from: https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html
18. Teras IC. Altera DE0 Boa. 2015. Available from: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=364
19. Maashri AA, DeBole M, Cotter M, Chandramoorthy N, Xiao Y, Narayanan V, Chakrabarti C. Accelerating neuromorphic vision algorithms for recognition. Proceedings of Third International Conference on Design Automation Conference (DAC 2012), ACM/EDAC/IEEE. 2012. p. 579–84.
20. Kaps J-P. Chai-Tea. Cryptographic hardware implementations of xTEA. Proceedings of the 9th International Conference on Cryptology in India: Progress in Cryptology; Kharagpur, India. 1484939: Springer-Verlag. 2008. p. 363–75.
21. Fan X, Gong G, Lauffenburger K, Hicks T. FPGA implementations of the Hummingbird cryptographic algorithm. Proceedings of the 9th International Conference on Hardware-Oriented Security and Trust (HOST 2010). 2010. p. 48–51.
22. Garcia R, Algredo-Badillo I, Morales-Sandoval M, Feregrino-Uribe C, Cumplido R. A compact FPGA-based processor for the Secure Hash Algorithm SHA-256. Computers and Electrical Engineering, 2014; 40(1):194–202.
23. Algredo-Badillo I, Morales-Sandoval M, Feregrino-Uribe C, Cumplido R. Throughput and efficiency analysis of unrolled hardware architectures for the SHA-512 Hash algorithm. (ISVLSI), 2012 IEEE Computer Society Annual Symposium on VLSI, IEEE, 10.1109/ISVLSI. 2012; 63:63–8.