Research Journal of Applied Sciences, Engineering and Technology 11(10): 1114-1123, 2015
DOI: 10.19026/rjaset.11.2126
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2015 Maxwell Scientific Publication Corp.
Submitted: June 24, 2015 Accepted: August 2, 2015 Published: December 05, 2015

Research Article

Secured Communication System Architecture Using Light Weight Algorithms

A. Prathiba and V.S. Kanchana Bhaaskaran
School of Electronics Engineering, VIT University, Chennai Campus, Chennai 600127,
Tamil Nadu, India

Abstract: Security of systems involved in the ubiquitous applications demands computationally less intensive operations. The proposed work aims at the design of the overall secured system architecture for the resource constrained platforms. The design architecture features; 1) the secured off-chip memory, 2) the master computation unit with a specialized encoding mechanism and 3) the encrypted data transfer to the slaves along with integrity and authentication. The design explores for the first time, the light weight cryptographic algorithms, both a Feistel cipher and a Substitution Permutation cipher (SPN), in its use for security compliance. Exhaustive analyses are made for the implementation results on the xc4VLX15-12sf363 Virtex FPGA. The time overheads for the memory encryption using the light weight Feistel cipher Tiny Encryption Algorithm (TEA) and the SPN cipher PRESENT are estimated for the actual 64-bit and the 16-bit custom derived implementations. The time latency for deriving the integrity and authentication extension to the slaves depicts nominal values acceptable for RFID communication applications.

Keywords: Authentication, confidentiality, integrity, light weight cryptography, system architecture

INTRODUCTION

The pervasive systems with limited processing capability, battery life and resources face many constraints in achieving the aspects of security, such as the confidentiality, the authentication, the data integrity and non-repudiation. The use of stronger cryptographic algorithms for increased system security may not be a viable solution for systems with the limitations identified above. The huge overhead incurred in the improvement of security can primarily penalize on the battery life time and the core area occupancy. In order to provide security for such embedded applications, a few light weight cryptographic algorithms such as the TEA, the PRESENT, the HIGHT, the mCRYPTON and the Humming Bird providing adequate security have been introduced (Wheeler and Needham, 1995; Bogdanov *et al*., 2007). Therefore, the researchers focus on the establishment of newer algorithms or suggestion of several modifications in the architectural parameters of the existing ones to suit the smart devices. In this context, the modifications of the widely adopted algorithms resulting in the DESL, DESXL have also been analyzed (Kitsos *et al*., 2012). The mechanisms involved in such solutions need to necessarily provide security in the communication with less power intensive operations. The security algorithms adopted for the RFID applications and embedded devices so, aimed at the selection of light weight algorithms and their optimal implementations.

The term light weight cryptography does not mean that it compromises on security. Rather it aims at providing an adequate security through an optimized implementation process, thus reducing the number of operational parameters, such as the number of rounds, reduced hardware through controlling the data path, energy and delay. References (Hanley and O'Neill, 2012; Kitsos *et al*., 2004; Tapalina, 2013) focus on various hardware and software algorithm implementations and their efficiency comparisons. Furthermore, the choice of the suitable algorithm for the pervasive applications has also been analyzed (Poschmann *et al*., 2007; Eisenbarth and Kumar, 2007). The literature concludes that the light weight cryptography results in the minimum usage of the available resources (Biham, 1997; Poschmann, 2009; Leander *et al*., 2007). Hence, the light weight algorithms can be preferred for such applications. They can trade off the properties, such as the speed, area and cost, against the two levels of:

- The extreme security to
- An adequate security

Such an approach can provide a dynamic balance between the security and the resources requirements there for.

The crypto processors hence demand effective solutions to cater to the present scenario (Ernest, 2014). The proposed work devises a secure architecture with a

Corresponding Author: A. Prathiba, School of Electronics Engineering, VIT University Chennai Campus, Chennai 600127, Tamil Nadu, India

specialized master computation unit design as a solution to the current application specific demands. Furthermore, an extension to the integrity check and the authentication mechanism are proposed by the hash value generation based on the results of the computation. In order to provide security, the light weight algorithms have been exploited in this study.

It devises a secure master-slave peripheral interface, which possesses the following characteristics:

- Secured data transfer from the external memory to the master computation unit.
- The master computation unit with a uniquely encoded 64-bit of data. The op-code, slave selection and operands 1 and 2, respectively are allotted with 16-bits each.
- Communication to the slaves with confidentiality, integrity check and authentication verification.

SYSTEMS SECURITY-THE CURRENT SCENARIO

System architecture normally has multiple custom made cores for the specific applications. The integration, interface, control and communication of the overall system are vulnerable to security threats. A typical block diagram of a system and its vulnerable regions for attacks on the communication bus are depicted in Fig. 1.

The following sub sections deal with the state of art of the memory protection and the authentication and integrity extension.

Memory encryption: Much of the information transfer involves the external memory transfers. Therefore, there is always a risk for the data in terms of transaction security, while moving from and to the memory. Hence, they are to be efficiently handled by adequate protection mechanisms. Such measures rely on the encryption schemes, which make the data unintelligible and the methodologies employed for protection can be addressed by the light weight schemes. A survey of the existing techniques of the memory encryption exposes three ways of memory protection, namely, the hardware enhancements, the operating system enhancements and the specialized industrial devices (Michael and Stephan, 2013). The overall focus of such memory encryption schemes is to provide:

- The security without any significant modification of the commodity processors
- No inordinate overhead in incorporating the security
- An effective solution in terms of the protection and promising better real time operating systems.

The traditional memory protection schemes employ the methods of encryption by several variations of AES cipher with one time pad encryption using the time stamp values. It is assumed that the private key required for the manipulation is stored in the trusted zone (Romain *et al*., 2009). A method of securing the programs by a secure cache structure is introduced to protect the off-chip memory through the use of the counter mode of AES (Huo *et al*., 2010). The off-chip data in the shared memory of multiprocessor systems is effectively analyzed by providing the data encryption and the authentication through data, Message Authentication Codes (MAC) and Initialization Vector (IV) combinations in Fangyong *et al*. (2013).

Authentication and data integrity: The traditional approach for the integrity and authentication of the communication relies on the set of promising solutions which can offer an extension to the encryption mechanism, even while not providing it as an additional and a complete standalone process. The one time pad scheme adopted in reference (Romain *et al*., 2009) validates that it is an effective way to achieve the authentication and integrity. Furthermore, the combinations of the techniques in terms of the authentication exchanges and the time stamps have resulted in various authentication schemes (Huo *et al*., 2010). The mechanism used for verifying the authentication utilizes the IV information as a tree of the history for the data transferred on the system bus in the time sequence (Fangyong *et al*., 2013). Some of the security enhancements by mutual authentication protocols are presented in Suna *et al*. (2014). The concept of the authenticated encryption demonstrates the use of a single algorithm with the same key for both

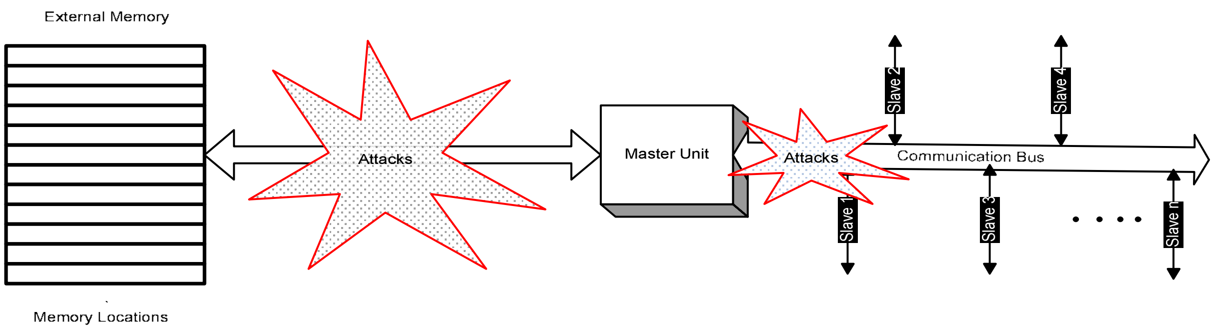


Fig. 1: Block diagram of security attacks on the system communication bus

the encryption and the authentication resulting in considerable area saving (Abdellatif *et al*., 2014). The I2C security protocol implemented in Jesús *et al*. (2011) provides the secure chip-to-chip encryption by the AES Galois counter mode and the authentication by the mutual exchange of handshakes. The secure architecture AEGIS combines the physical random functions, hash and IV values in order to satisfy the system security requirements (Suh *et al*., 2007).

To summarize, the system security normally is provided with the following:

- An efficient dynamic key management
- Trusted and non-trusted zone placement of the blocks in the architecture
- The memory access protection
- The address remapping schemes

The latency incurred for the additional memory protection can result in reduced throughput of the system. Hence, the identification of a good balance between the operational scheduling and the sequencing can result in a better system security without majorly degrading the speed performance. The establishment of authentication and integrity check through extension of the confidentiality mechanism employed offers a better solution.

The proposed design of the secured system architecture addresses both the above factors with a unique instruction bit format and the dynamic handling of integrity and authentication at the expense of lesser energy consumption.

THE PROPOSED ARCHITECTURE

The proposed solution to offer the security employs a full memory encryption using the light weight encryption algorithms, namely, TEA/PRESENT. The work assumes that the keys for the full memory protection lie in the trusted zone and the encryption of the full memory is done within the trusted boundary. Hence, a chain of trust from the information inputs to the computation unit is considered as established through the secured information transfer from the protected external memory. The vulnerability threat on the communication interface between the memory and the master unit is designed to be protected by the encrypted data transfer in the proposed architecture. The schematic depicting the arrangement of the external memory, the master computation unit and the slaves are shown in Fig. 2.

A block size of 64-bits is chosen to match the parametric block size of the light weight algorithms employed for the security. The proposed architecture stores the encrypted input data in the memory. The encrypted data from the memory is decrypted and decoded appropriately as the 16 bits operation code, viz. 16 bits each for the two operands and the 16 bits slave selection line. The computations are done in the master unit. The result of computation will be encrypted by the custom derived 16-bit TEA/PRESENT algorithm. The encrypted output will be communicated to the slaves. The master computational unit architecture is as shown in Fig. 3. The 64 bit instruction bit format showing the unique bit details are shown in Fig. 4. The hash value is generated as a digest of the selection line, the key and the actual plain text of the result. Figure 5 shows the hash value generation to ensure the data integrity between the master and the slave.

The 64 bit instruction format shown in Fig. 4 has the individual details as follows:

The sequences of processes occurring in the system are detailed below. Note that the response and/or reasoning of each step are provided in italics:

External memory write operation:

Step 1: Read the 64 bit Information
Step 2: Encrypt the 64-bit data using the light weight TEA/PRESENT.
Step 3: Store the cipher data in the specified memory location.

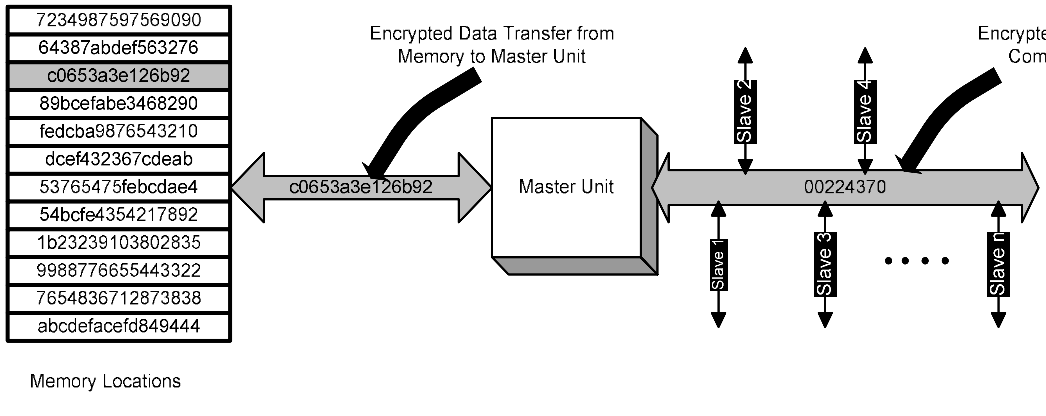


Fig. 2: Secured data transfer on the bus

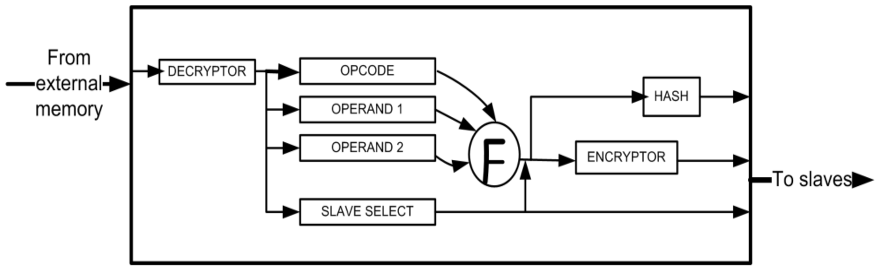


Fig. 3: Master computational unit architecture

Opcode Size	Slave Select Lines	Operand 1	Operand 2
16 bits	16 bits	16 bits	16 bits

Fig. 4: The 64-bit instruction format

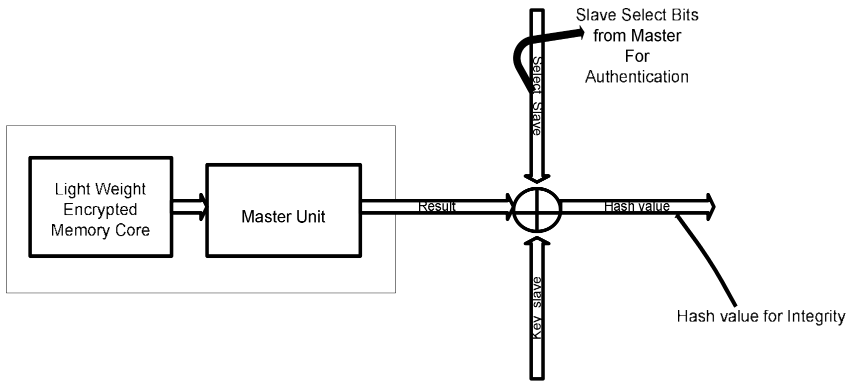

Slave Select Bits from Master For Authentication
Select Slave
Light Weight Encrypted Memory Core
Master Unit
Result
Hash value
Key slave
Hash value for Integrity

Fig. 5: Proposed hash value generation for integrity check

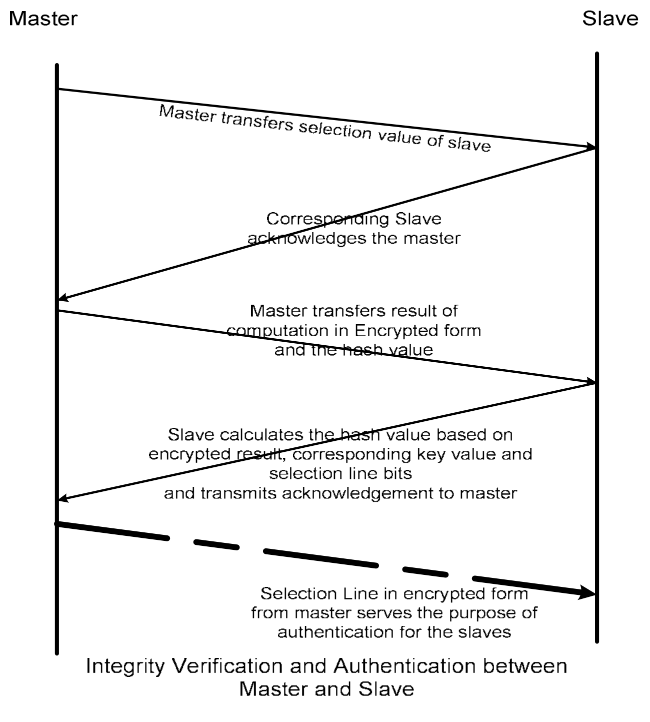

Master
Slave
Master transfers selection value of slave
Corresponding Slave acknowledges the master
Master transfers result of computation in Encrypted form and the hash value
Slave calculates the hash value based on encrypted result, corresponding key value and selection line bits and transmits acknowledgement to master
Selection Line in encrypted form from master serves the purpose of authentication for the slaves
Integrity Verification and Authentication between Master and Slave

Fig. 6: The integrity verification and the authentication exchange

Table 1: Information flow in the system

Assigned task	Data input in hexadecimal representation	Details of operation
MASTER OPERATION		
64 bit plain text to be stored in encrypted form in the respective external RAM location	0000_0000_0011_0011	16 bit instruction code-0000, 16 bit Slave Select-0000, 16 bit operand 1-0011, 16 bit operand 2-0011
64 bit Cipher Text in the external RAM location at the specified address	f34c_6ce6_b294_22d0	64 bit encrypted data
Decrypted and decoded information in the master unit	Operation code :0000 (AND) Slave Select:0000 (Slave 1) Operand 1:0011 Operand 2:0011	64 bit decoded data
Computation output of the master unit in original form	0011	16 bit plain output result
Encrypted output from master unit to slave 1	0070	16 bit encrypted result
Select value sent to slave	0000	Slave selection line
Hash value sent to slave	0012	Hash of select line, corresponding key and the plain form of result
SLAVE OPERATION		
Integrity check and Authentication verification by the slaves	0000	Slave Selection line
	0070	Encrypted output
	0012	Hash value
	0011	Decrypted output

Master operation:

Step 1: Read the 64 bit encrypted data from the desired address.

Step 2: Decrypt and decode the 16 bit each instruction code, slave select and the operands.

Step 3: Perform the computation as specified by the instruction code.

Step 4: Encrypt the 16 bit result of the computation unit by using the appropriate light weight with block size as 16.
This will provide confidentiality of the data

Step 5: Transmit the slave selection line along the bus and wait for the acknowledgement from the corresponding slave.
This step will ensure that selection of slave is done by the authenticated master.

Step 6: Transmit the confidential data along the communication line and also transmit the generated hash value.
The data integrity is ensured by this process.

Slave operation:

Step 1: Based on the selection line from the master, the slave response will be sent as an acknowledgement to the master.

Step 2: The transmitted and secured result will be decrypted and the hash values are calculated.

Step 3: The comparison operation between the hash values and the value sent by the master will be performed. A positive or a negative acknowledgement, as per the result of the comparison will be sent to the master along with the selection line of the respective slave.

The protocol exchange to verify the authentication and the integrity check is shown in Fig. 6. The information flow in the system for the AND operation of the two 16 bit data is given in Table 1.

THE SYSTEM ARCHITECTURE: DISCUSSION

The memory encryption latency adds overhead to the fetch-decode-execute phases of the processing unit. Since the proposed design encapsulates the instruction, the slave selection lines and the data as single block information, the latency is bound to be minimal when compared to the encryption of each of the 16 bit information. The encryption/decryption time is measured to be 157.77ns for the 64-bit implementation and 128.41ns for the 16-bit TEA. The values pertaining to the PRESENT are 50.07ns and 45.76ns respectively. Hence, it can be observed that this does not result in significant degradation in the system speed performance. Though the one-to-one mapping of the encrypted and the unencrypted information persists, the details of the individual information units are not decodable. This is due to the fact that the mapping is designed with a unique architecture. For the anonymous attacker, the possibility of decoding the appropriate information unit is made infeasible by the specific decoding design incorporated in the proposed architecture. The master unit can perform a total of 2^{16}

possible operations and it can also communicate up to 2^{16} unique slaves.

The hash value used for the integrity check between the master computation unit and the slaves in this approach depends on the results of computation and this factor makes the system resistant to attacks. Furthermore, any attempt to provide modification on the information unit results in the incorrect selection of slaves by the master and it is easily detected by the respective slaves through their handshake exchanges with the master. The slave will retrieve the appropriate information after the decryption of the customary data from the master.

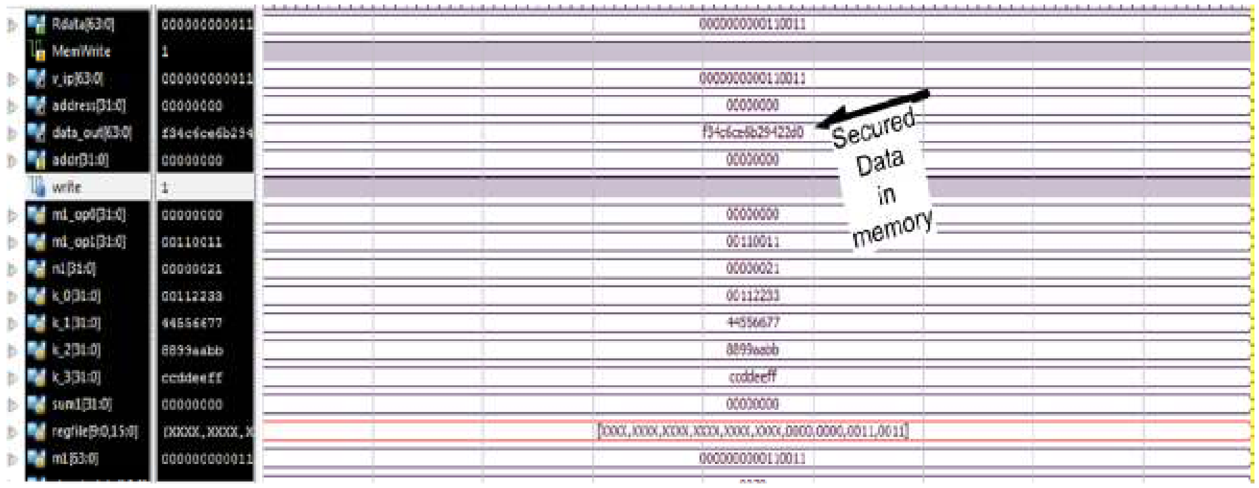
The proposed solution for the authentication and the integrity is accomplished as an extension of the encryption process. Furthermore, this solution provides a light weight means to facilitate these goals of cryptography without any additional high computation requirements, which suits the resource constrained environments. Each slave will have16 bit decryption unit for decrypting the result.

The simulation results of the various information flow paths are depicted in Fig. 7 to 10.

CASE STUDY

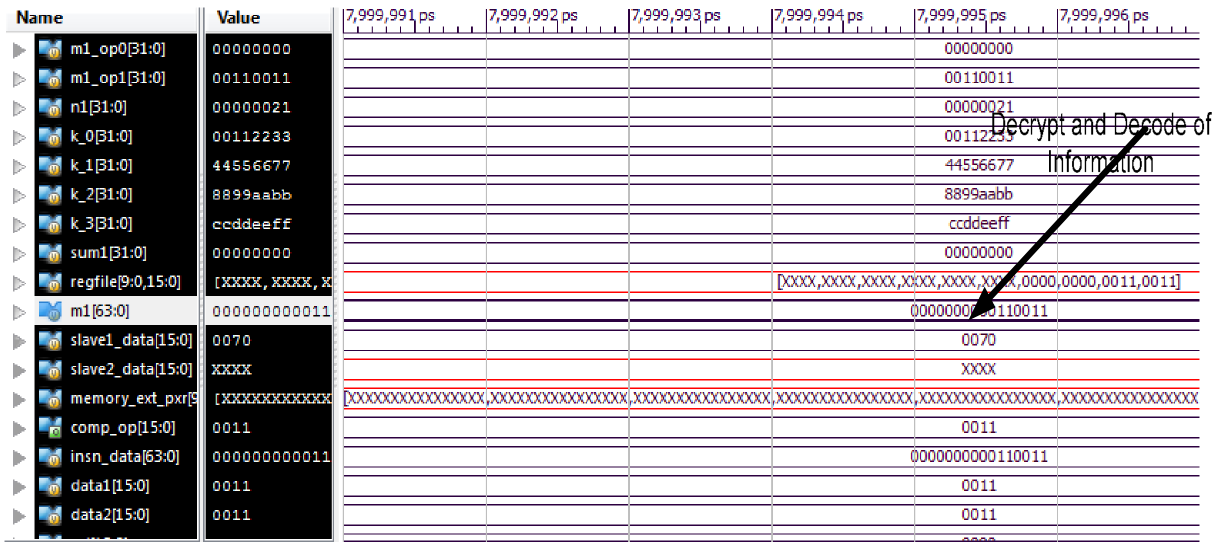
A Feistel cipher divides the entire data into two halves and works on each half independently, while the block cipher works on the entire block as a whole.
Brief overviews of the algorithms are given below:

Tiny encryption algorithm:
The light weight Feistel cipher: The Tiny Encryption Algorithm (TEA) developed at the Cambridge



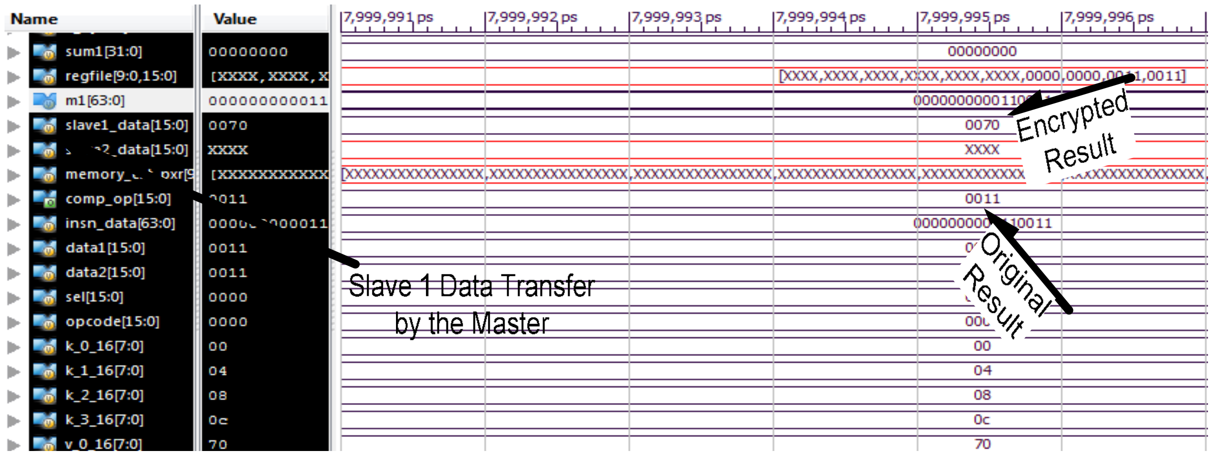
```
Rdata[63:0]        000000000011    0000000000110011
MemWrite           1
v_ip[63:0]         000000000011    0000000000110011
address[31:0]      00000000        00000000
data_out[63:0]     f34c6ce6b294    f34c6ce6b29422d0
addr[31:0]         00000000        00000000
write              1
m1_op0[31:0]       00000000        00000000
m1_op1[31:0]       00110011        00110011
n1[31:0]           00000021        00000021
k_0[31:0]          00112233        00112233
k_1[31:0]          44556677        44556677
k_2[31:0]          8899aabb        8899aabb
k_3[31:0]          ccddeeff        ccddeeff
sum1[31:0]         00000000        00000000
regfile[9:0,15:0]  [XXXX,XXXX,X    [XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,0000,0000,0011,0011]
m1[63:0]           000000000011    0000000000110011
```

Fig. 7: Encrypted information as stored in the external RAM



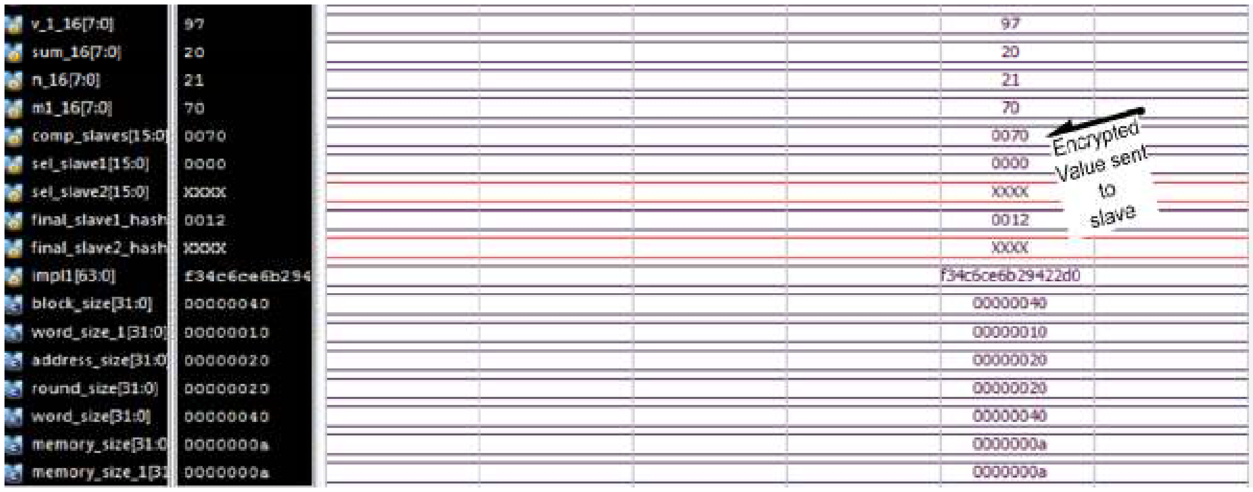
```
Name                    Value
m1_op0[31:0]            00000000           00000000
m1_op1[31:0]            00110011           00110011
n1[31:0]                00000021           00000021
k_0[31:0]               00112233           00112233
k_1[31:0]               44556677           44556677
k_2[31:0]               8899aabb           8899aabb
k_3[31:0]               ccddeeff           ccddeeff
sum1[31:0]              00000000           00000000
regfile[9:0,15:0]       [XXXX,XXXX,X       [XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,0000,0000,0011,0011]
m1[63:0]                000000000011       0000000000110011
slave1_data[15:0]       0070               0070
slave2_data[15:0]       XXXX               XXXX
memory_ext_pxr[9        [XXXXXXXXXXXX      [XXXXXXXXXXXXXXXX,XXXXXXXXXXXXXXXX,XXXXXXXXXXXXXXXX,XXXXXXXXXXXXXXXX,XXXXXXXXXXXXXXXX,XXXXXXXXXXXXXXXX
comp_op[15:0]           0011               0011
insn_data[63:0]         000000000011       0000000000110011
data1[15:0]             0011               0011
data2[15:0]             0011               0011
```

Fig. 8: Decryption and the decoding of information in the master unit



```
Name                 Value
sum1[31:0]           00000000           00000000
regfile[9:0,15:0]    [XXXX,XXXX,X       [XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,0000,0000,00 1,0011]
m1[63:0]             000000000011       00000000000110
slave1_data[15:0]    0070               0070
s   2_data[15:0]     XXXX               XXXX
memory_    oxr[9     [XXXXXXXXXXX       [XXXXXXXXXXXXXXX,XXXXXXXXXXXXXXX,XXXXXXXXXXXXXXX,XXXXXXXXXXXXXXX,XXXXXXXXXXX  XXXXXXXXXXXXXXX,
comp_op[15:0]        0011               0011
insn_data[63:0]      0000   000011      000000000  10011
data1[15:0]          0011               0
data2[15:0]          0011
sel[15:0]            0000
opcode[15:0]         0000               00
k_0_16[7:0]          00                 00
k_1_16[7:0]          04                 04
k_2_16[7:0]          08                 08
k_3_16[7:0]          0c                 0c
v_0_16[7:0]          70                 70
```

Fig. 9: Operation on data and transfer of encrypted data_16 to respective slaves



```
v_1_16[7:0]          97                 97
sum_16[7:0]          20                 20
n_16[7:0]            21                 21
m1_16[7:0]           70                 70
comp_slaves[15:0]    0070               0070
sel_slave1[15:0]     0000               0000
sel_slave2[15:0]     XXXX               XXXX
final_slave1_hash    0012               0012
final_slave2_hash    XXXX               XXXX
impl1[63:0]          f34c6ce6b294       f34c6ce6b29422d0
block_size[31:0]     00000040           00000040
word_size_1[31:0]    00000010           00000010
address_size[31:0]   00000020           00000020
round_size[31:0]     00000020           00000020
word_size[31:0]      00000040           00000040
memory_size[31:0]    0000000a           0000000a
memory_size_1[31     0000000a           0000000a
```

Fig. 10: Authentication and the message integrity check for the slave

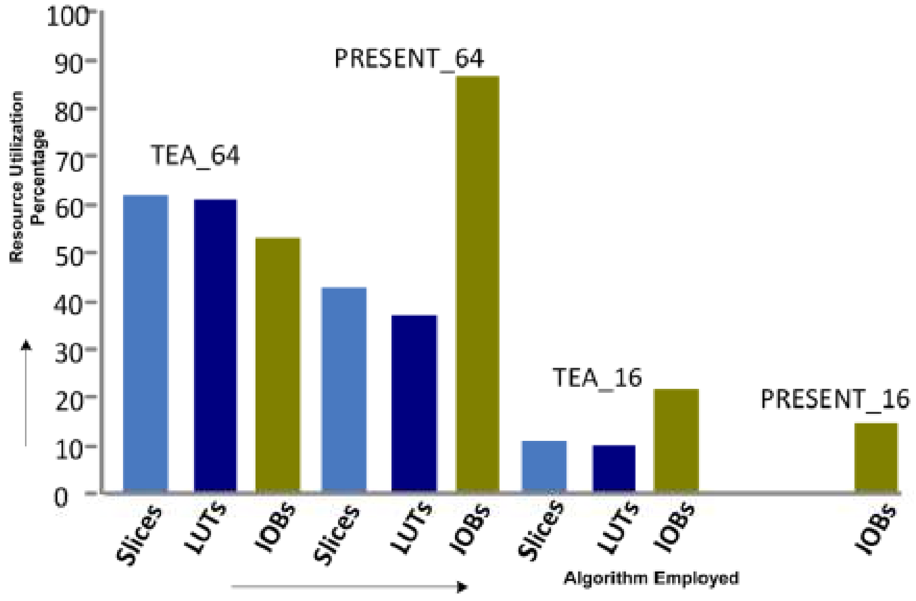


Fig. 11: Area utilization of the TEA/PRESENT encryption/decryption core for xc4VLX15-12sf363

University in UK with the main requirements of being an extremely small and a strong algorithm. It is a 64-bit symmetric block cipher of Feistel type and it uses a 128 bit key. This performs 32 rounds of defined collection of operations. It may be noted that even a minimum of 6 round is sufficient to provide adequate confusion and the diffusion of data. It incurs negligible setup time and no preset or the initialization phase is required. An additional advantage of this Feistel type routine is that it relies solely on the XOR and the ADD logic operations to provide the non-linearity. The other primary operation involved was the shift operation.

PRESENT: The light weight substitution permutation cipher: The PRESENT algorithm is a Substitution Permutation type symmetric block cipher with a block size of 64 bits and the key variants of 80 and 128 bits. The algorithm was proposed for its hardware implementation efficiency. It has 31 rounds and the operations involved are add-round key, substitution and permutation. The non-linearity is handled largely by the 4×4 substitution boxes employed in the algorithm. The proposed work utilizes the 80 bit key version.

RESULTS AND DISCUSSION

Table 2 shows the device utilization results of the system when implemented in the FPGA xc4VLX15-12sf363. It can be observed from Fig. 11 that the resource utilized for implementing the 64 bit encryption/decryption version exhausts only a maximum of 60% of the total resource availability and hence sufficient additional resources are available for the rest of the system modules. The slave has 16 bit decryption cores for extracting the result. Figure 12 shows the time incurred for the encryption using the different block sizes of the TEA and PRESENT algorithms. Figure 13 depicts the encryption time comparison for plain and secured RAM. Optimization mechanisms namely, parallelization and pipelining are bound to improve the system security without impacting the speed performance of the system.

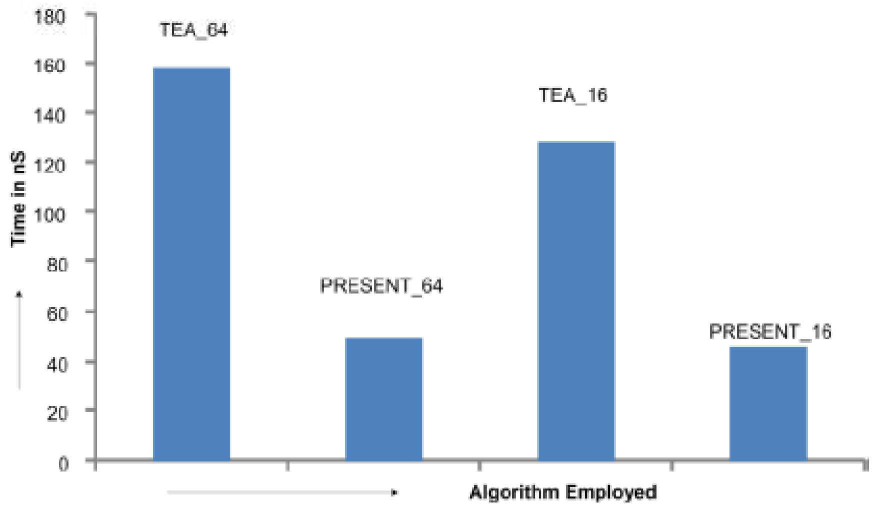


Fig. 12: Latency in ns of TEA, PRESENT for different block sizes for xc4VLX15-12sf363

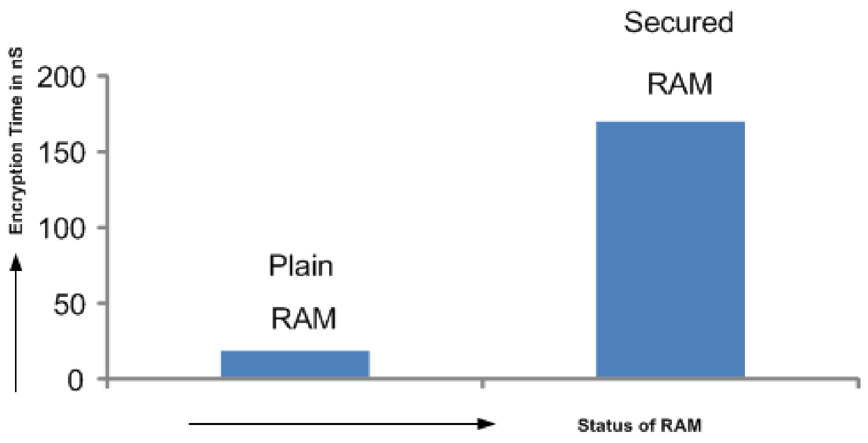


Fig. 13: Latency without and with memory encryption for xc4VLX15-12sf363

Table 2: Implementation results of the TEA and PRESENT encryption/decryption for various block sizes for xc4VLX15-12sf363

Algorithm employed	Resource type	Utilization ratio
TEA Block size 64 bits	No of Slices	3820/6144
	No of 4 input LUTs	7554/12288
	No of Bonded IOBs	128/240
TEA Block size 16 bits	No of Slices	486/6144
	No of 4 input LUTs	864/12288
	No of Bonded IOBs	32/240
PRESENT Block size 64 bits	No of Slices	2688/6144
	No of 4 input LUTs	4616/12288
	No of Bonded IOBs	209/240
PRESENT Block size 16 bits	No of Slices	726/6144
	No of 4 input LUTs	1252/12288
	No of Bonded IOBs	53/240

The ISO standard for the contactless smart cards defines the operating frequency as 13.56 MHz. The delay incurred in providing the authentication and the integrity verification in the proposed work is 7.275ns. The throughput values are estimated using the expression (block size*frequency)/delay. The typical value obtained for the throughput value is 651bits per second. Hence, it may be noted that the faster authentication and verification processes of the proposed system, will be highly preferable for systems involving such throughput requirements.

CONCLUSION AND RECOMMENDATION

The proposed work aims at the secured system architecture design which will offer data confidentiality, authentication and message integrity. The authentication and integrity is realized as an extension of the confidentiality mechanism. The time overhead for storing the secured data in memory using the light weight Feistel Cipher Tiny Encryption Algorithm (TEA) is found to be 157.77ns for the 64-bit implementation and 128.41ns for the 16-bit implementations, respectively. The results obtained for the light weight SPN cipher PRESENT algorithm for the 64-bit and the 16-bit realizations are found to be 50.07ns and 45.76ns, in that order. The computation unit employed utilizes the custom-derived 16-bit block size security cipher for the secured transfer of the results of computation on the communication bus to the slaves. Thus, the proposed architecture stands validated in providing an adequate system security to the resource constrained platforms.

The individual approaches for arriving at different forms of the security in the proposed solution can be adopted as per the existing security architectures, without incorporating any major modifications. The complex key management techniques or on-the-fly key generation mechanisms can also further improve the security level of the proposed architecture, which is being planned as the extended work. Furthermore, the unused operational codes can be used to zero-ize the keys, whenever a security breach is detected.

REFERENCES

Abdellatif, K.M., R. Chotin-Avot and H. Mehrez, 2014. Authenticated encryption on FPGAs from the static part to the reconfigurable part. Microprocess. Microsy., 38(6): 526-538.

Biham, E., 1997. A fast new DES implementation in software. In: Biham, E. (Ed.), FSE 1997. LNCS 1267, Springer, Heidelberg, pp: 260-272.

Bogdanov, A., L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin and C. Vikkelsoe, 2007. PRESENT: An ultra-lightweight block cipher. Proceeding of the 9th Internationl Workshop Cryptographic Hardware and Embedded Systems (CHES'07), 4727: 450-466.

Eisenbarth, T. and S. Kumar, 2007. A survey of lightweight-cryptography implementations. IEEE Des. Test Comput., 24(6): 522-533.

Ernest, W., 2014. Locking Down the Chip. Semiconductor Engineering. Retrieved from: http://semiengineering.com/locking-down-the-chip-2/.

Fangyong, H., H. Hongjun, X. Nong and L. Fang, 2013. Bus and memory protection through chain-generated and tree-verified IV for multiprocessors systems. Future Gener. Comp. Sy., 29(3): 901-912.

Hanley, N. and M. O'Neill, 2012. Hardware comparison of the ISO/IEC 29192-2 block ciphers. Proceeding of the IEEE Computer Society Annual Symposium on VLSI, pp: 57-62.

Huo, W.J., Z.L. Liu and X.C. Zou, 2010. PEM: A lightweight program memory encryption mechanism for embedded processor. J. China Univ., Posts Telecomm., 17(1): 77-84.

Jesús, L. A. Armando, Z. Aitzol, B. Unai and J. Jaime, 2011. I2CSec: A secure serial Chip-to-Chip communication protocol. J. Syst. Architect., 57(2): 206-213.

Kitsos, P., N. Sklavos, M.D. Galanis and O. Koufopavlou, 2004. 64-bit Block ciphers: Hardware implementations and comparison analysis. Comput. Electr. Eng., 30(8): 593-604.

Kitsos, P., N. Sklavos, M. Parousi and A.N. Skodras, 2012. A comparative study of hardware architectures for lightweight block ciphers. Comput. Electr. Eng., 38(1): 148-160.

Leander, G., C. Paar, A. Poschmann and K. Schramm, 2007. New lightweight DES variants. In: Biryukov, A. (Ed.), FSE, 2007. LNCS 4593, Springer, Berlin, Heidelberg, pp: 196-210.

Michael, H. and T. Stephan, 2013. Memory encryption: A survey of existing techniques. Dartmouth Technical Report TR13-001.

Poschmann, A.Y., 2009. Lightweight cryptography: Cryptographic engineering for a pervasive world. Ph. D. Thesis, Ruhr-University Bochum, Germany.

Poschmann, A., G. Leander, K. Schramm and C. Paar, 2007. New light-weight crypto algorithms for RFID. Proceeding of the IEEE International Symposium on Circuits and Systems (ISCAS, 2007). New Orleans, LA, pp: 1843-1846.

Romain, V., G. Guy, D. Jean-Philippe, W. Eduardo, T. Russell and B. Wayne, 2009. A security approach for off-chip memory in embedded microprocessor systems. Microprocess. Microsy., 33(1): 37-45.

Suh, G.E., C.W. O'Donnell and S. Devadas, 2007. Aegis: A single-chip secure processor. IEEE Des. Test Comput., 24(6): 570-580.

Suna, C., K. Hyunseok, L. Sangyeon, L. Kangbok and L. Heyungsub, 2014. A fully integrated CMOS security-enhanced passive RFID tag. ETRI J., 36(1): 141- 150.

Tapalina, B., 2013. LICRYPT: Lightweight cryptography technique for securing smart objects in internet of things environment. Article, CSI Communications, May 2013.

Wheeler, D. and R. Needham, 1995. TEA, a tiny encryption algorithm. In: Preneel, B. (Ed.), FSE 1994. LNCS 1008, Springer, Heidelberg, pp: 363-366.