ISSN 2047-3338

Developing Configurable Security Algorithms for Embedded System Storage

Sunil D. Bobade[1] and Vijay R. Mankar[2]
[1]S.G.B. Amravati University, India
[2]MSBTE, Pune, India

Abstract– **The constrained operating environments of many FPGA based embedded systems require flexible security that can be configured to minimize the impact on FPGA area and power consumption. To ensure the overall security of embedded systems with off-chip memories, it is essential to safeguard the confidentiality and integrity of the data that travels between the system-on-chip part of the embedded system and the off-chip memory. This paper proposes a complete hardware solution for embedded systems that fully protects off-chip memory. The confidentiality and integrity is achieved by using a modified Elliptic Curve Cryptographic (ECC) approach. Initially, with respect to the instruction from the processor, the security level for the address is identified and based on the security level the encryption process follows, thereby storing the data in the form of cipher text in the external memory location. During read operation, based on the address, the segment id is verified and then the decrypted data is allowed to the processor after appropriate verification. These modules safeguard external memories for embedded processors against a series of well-known attacks, including replay attacks, spoofing attacks and relocation attacks. The complete module is realized using Xilinx ISE 14.1 software using verilog coding with a target device as virtex-4 xc4vlx200-11-ff1513. The functional verification of the design is done by simulating the design.**

Index Terms– **Elliptic Curve Cryptographic, Embedded System, VLSI, Xilinx, Confidentiality and Integrity**

I. INTRODUCTION

SYSTEM security is an increasingly important design criterion for many embedded systems. These systems are often portable and more easily attacked than traditional desktop and server computing systems. FPGAs are quickly becoming ubiquitous components in many low-cost embedded systems. These systems often contain little more than an FPGA, external memory, I/O ports and interfaces to sensors and monitors [1]. In addition to standard concerns regarding system performance and power consumption, security has become a leading issue for many embedded applications [2]. Although the programmed contents of FPGAs are frequently protected with bitstream encryption, external memory transfers can easily be observed and may reveal important application information. Some memory protection can be provided by simply encrypting data prior to external memory transfer. The confidentiality and integrity of sensitive information is a critical component of any secure system [3]. Typically, this protection is implemented at least in part through the use of symmetric algorithms, like DES, AES, or countless others.

The external memory of an embedded system can face a variety of attacks Well-known attacks include spoofing, relocation, and replay attacks [4]. A spoofing attack (Figure 1) occurs when an attacker provides a random data value on the bus, causing the system to malfunction. A relocation or splicing attack (Figure 2) occurs when an instruction put on the bus by an attacker is copied from a different bus address. If the whole memory is encrypted with the same key, the swapped instruction will be executed instead of the original instruction. For example, a swapped instruction could make the program jump to malicious code stored in a non-ciphered part of memory. The last type of attack a system might face is a replay attack (Figure 3). This attack is similar to a relocation attack but an attacker provides a data value that was previously located at an address before it was overwritten.

In this paper, a new FPGA-based approach is proposed which provides confidentiality and integrity to off-chip data accesses made by a processor on the FPGA chip. A hardware-based security core determines the appropriate data security level as memory accesses occur in conjunction with an embedded real-time operating system. Proposed approach targets replay, relocation, and spoofing attacks. Confidentiality in proposed system utilizes the ECC crypto processor [5] designed using area efficient Double point multiplication algorithm [6] designed in our preliminary work. In proposed implementation, a time stamp value (TS), data, data and address of the write data are used as input to an ECC encryption circuit to generate the cipher text, which in turn is stored in the external memory. Rather than encrypting write data directly, as in conventional techniques[7]-[8], our approach generates an encrypted data by subjecting a specific format of data, memory address and the TS value to the ECC module during each write operation. The Time stamp value considered here is the private key to the ECC module. The use of time stamps and data addresses in our approach protects read/write data against replay and relocation attacks [9].

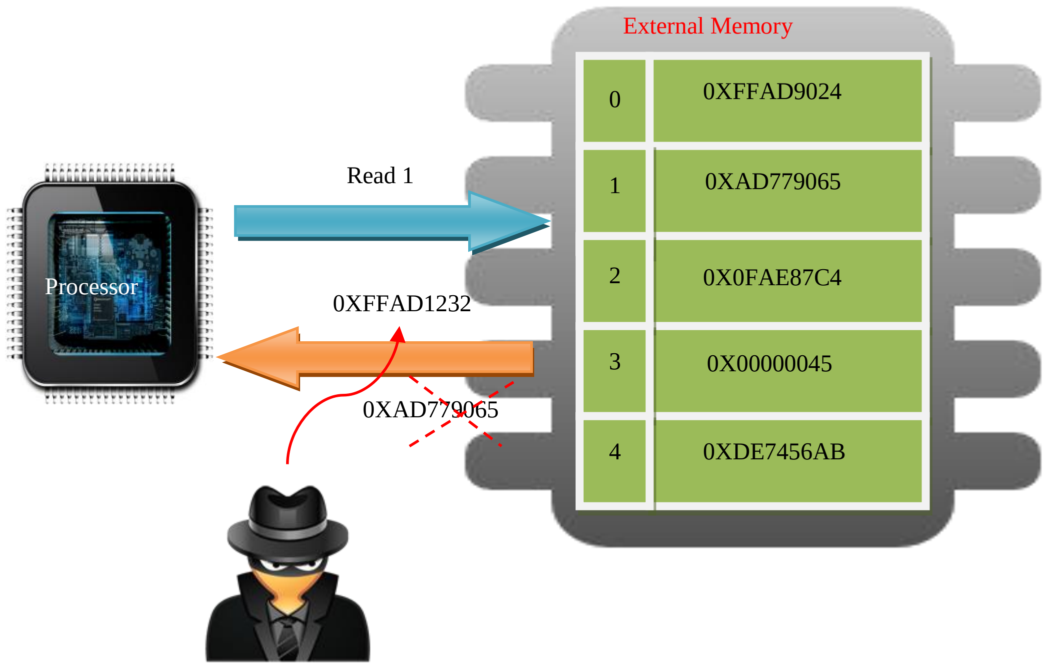

Figure 1: Spoofing attack

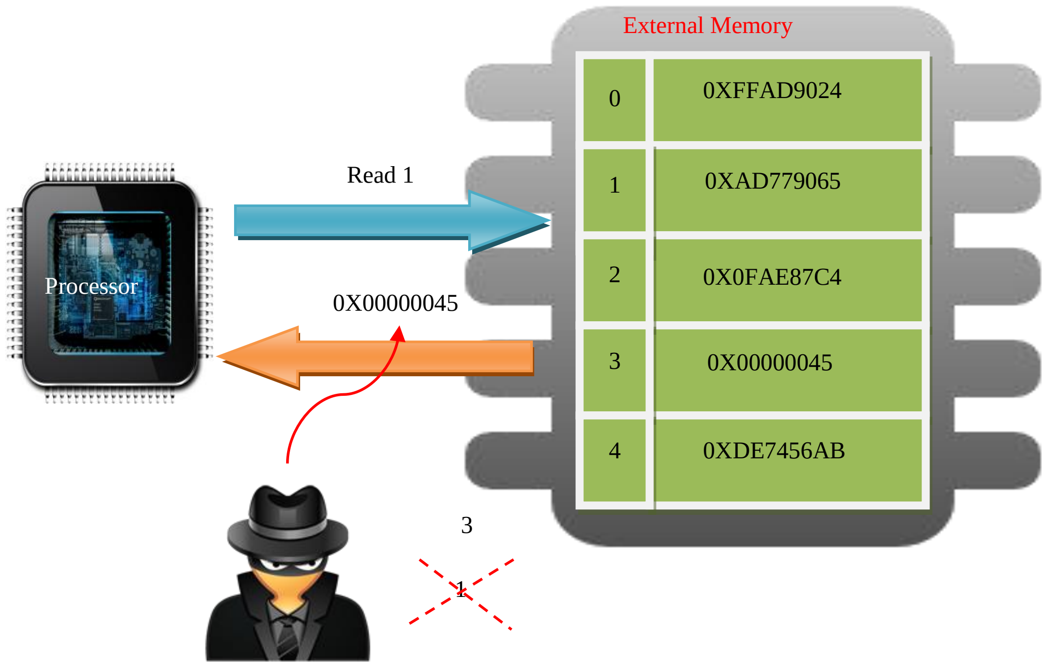

Figure 2: Relocation attack

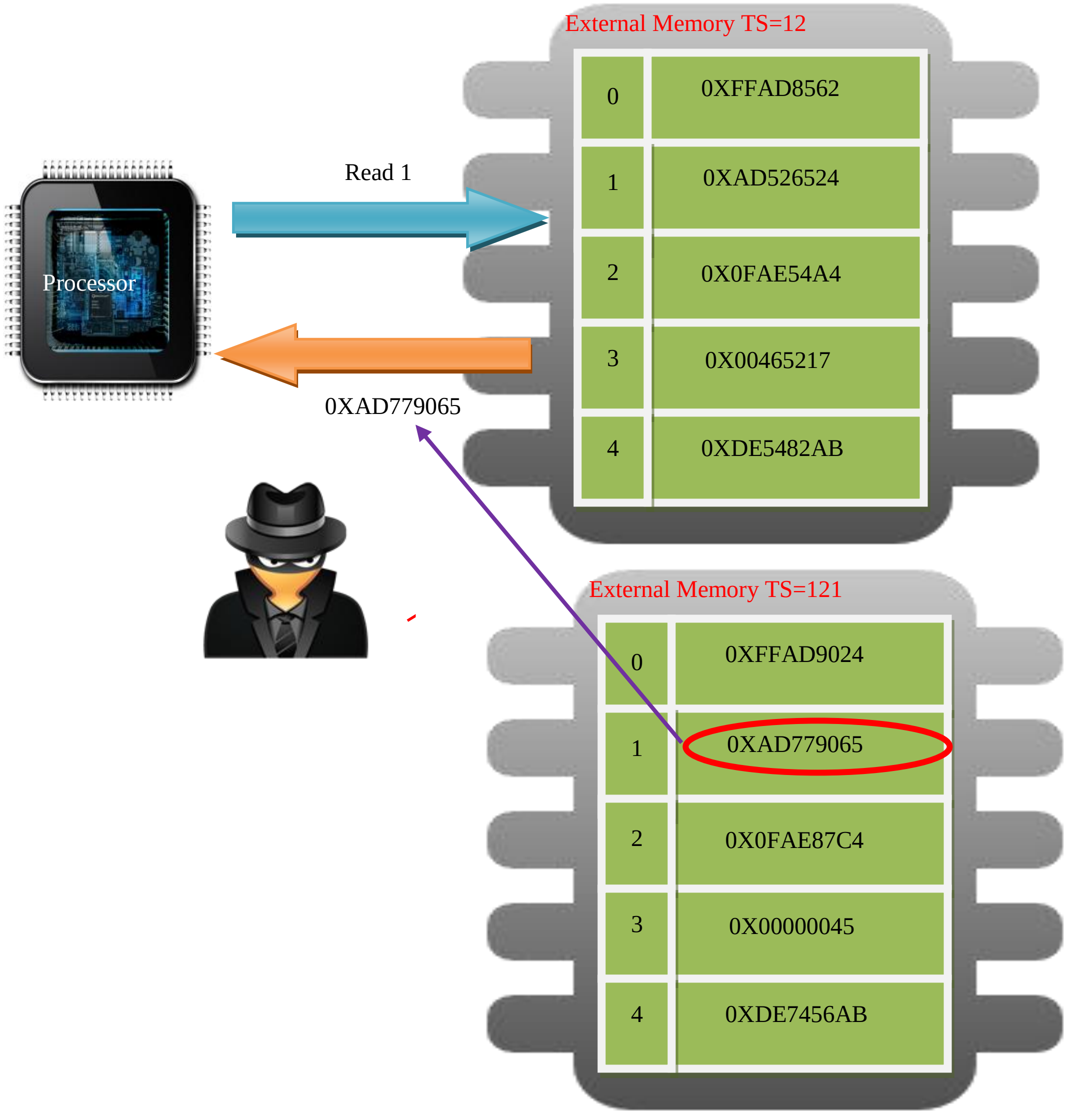

Figure 3: Replay attack

If a data value is replayed, the TS used for ciphering will differ from the one used for deciphering. If a data value is relocated, its address will differ from the one used to generate the cipher text. In both cases, the deciphered data will be invalid. Our FPGA-based security core provides both confidentiality and integrity for data stored externally to an FPGA which is accessed by a processor on the FPGA chip. The complete system is synthesized and simulated using Xilinx ISE. Power report, Device utilization summary and performance of our system is designed and compared with the similar works reported recently.

The rest of the paper is structured as follows: Related work is described in Section II. Research methods and motivation for our work is described in Section III. Details of our proposed hardware security core are explained in Section IV. A description and analysis of experimental results are provided in Section V. Section VI concludes the paper.

II. RELATED WORKS

In [10], Satyajeet Nimgaonkar *et al.* presented a novel energy efficient Memory Integrity Verification scheme for embedded systems with the focus on achieving energy efficiency in cryptographic Memory Integrity Verification (MIV) mechanism. The work contributed with a novel energy efficient approach called Timestamps Verification (TSV) to provide Memory Integrity Verification in embedded systems. The research also elaborated the new technique along with its theoretical evaluation, simulation results, and experimental evaluation. The results proved that the energy savings in the TSV approach are in the range of 36–81% when compared with traditional MIV mechanisms.

Jeremie Crenne *et al.* in [11] presented a new approach to embedded system security focusing on the protection of application loading and secure application execution. During secure application loading, an encrypted application was transferred from on-board flash memory to external double data rate synchronous dynamic random access memory (DDR-SDRAM) via a microprocessor. Following application loading, the core-based security technique provides both confidentiality and authentication for data stored in a microprocessor's system memory. The benefits of the low overhead memory protection approaches were demonstrated using four applications implemented in a field-programmable gate array (FPGA) in an embedded system prototyping platform. Each application required a collection of tasks with varying memory security requirements. The configurable security core implemented on-chip inside the FPGA with the microprocessor allowed for different memory security policies for different application tasks. An average memory saving of 63% was achieved for the four applications versus a uniform security approach. The lightweight circuitry included to support application loading from flash memory adds about 10% FPGA area overhead to the processor-based system and main memory security hardware.

Ma Haifeng *et al.* in [12] introduced an efficient scheme to protect data confidentiality and integrity. Based on the local characteristic of data accessing, the scheme set different counter length for memory area according to different accessing frequencies and the counter length can be dynamically adjusted. The analysis and the simulation results indicated that compared with the counter mode encryption, the scheme can decrease memory space overhead and the number of overflow. The proposed scheme can be applied to other schemes of protecting confidentiality and integrity based on counters and can satisfy performance requirements for most applications.

A novel energy efficient approach for MEMory integrity Detection and Protection (MEM-DnP) was presented by Satyajeet Nimgaonkar *et al.* in [13]. The key feature of MEM-DnP was that it can be adaptively tune a memory integrity verification module by using a sensor module. This significantly reduced the energy overheads imposed on an embedded system as compared to the conventional memory integrity verification mechanisms. The simulation results showed that the average energy saved in the combined detection and protection mechanism ranges from 85.5% to 99.998%. This was substantially higher compared to the results achieved in base case simulations with traditional memory integrity verification techniques.

In [14], Bao Liu and Brandon Wang presented a reconfigurable reversible computing-based cryptography, and a generic reconfiguration-based VLSI design-for-security methodology. In the case studies based on a SPARC V8 LEON2 processor, the new technique prevented software- or hardware-based code injection attacks at cost of 0.72% area increase, negligible power consumption increase and no performance degradation; it also further prevented a hardware Trojan from gaining unauthorized memory access at cost of 4.42% area increase, negligible power consumption increase, and 11.30% critical path delay increase. Apostolos P. Fournaris and Nicolas Sklavos [15] provided an overview from ES hardware perspective of methods and mechanisms for providing strong security and trust. The various categories of physical attacks on security related embedded systems were presented along with countermeasures to thwart them and the importance of reconfigurable logic flexibility, adaptability and scalability along with trust protection mechanisms is highlighted. Those mechanisms were adopted in order to propose a FPGA based embedded system hardware architecture capable of providing security and trust along with physical attack protection using trust zone separation. The benefits of such approach were discussed and a subsystem of the proposed architecture is implemented in FPGA technology as a proof of concept case study. In [16] Zhenglin Liu *et al.* provided an approach for both data confidentiality and integrity authentication and enable a good tradeoff between on-chip memory cost, performance overhead and security. The proposed double-layer protection mechanism works effectively on the occasions where on-chip memory overhead was sensitive and the speed requirement is not so high. Comparing with tree based approaches, such as AEGIS, their approach has a great advantage in terms of performance cost, but the on-chip memory overhead is relatively larger than that of tree based approaches.

III. RESEARCH METHODOLOGY

There are two ways to include security primitives to the device. The first one is to try to add software protections. But due to the restricted capabilities of embedded systems, software approach is not recommended due to the processing overhead. The second solution is a more practical. The idea is to add some hardware primitives which are running on the fly and checking the system integrity. Energy is a critical point as soon as embedded systems are targeted especially when they are autonomous. Hardware architectures provide the most interesting solutions from an energy efficient point of view. For example, a hash algorithm consumes 80 times more energy in software than a hardware implementation. Here the goal is to provide an efficient hardware solution, fully transparent for the user (no system stall) and also for the software designer who builds application for the device. The hardware primitives must take care of the security primitive to guarantee the data confidentiality and the system integrity to prevent any unauthorized software modification. The challenge is to offer area efficient security solution.

IV. PROPOSED METHOD

The proposed security technique for memory security by utilizing a Hardware security core, which is realized using FPGA logic along with an embedded memory, is depicted in figure. The proposed hardware security core is designed with an intention in order to switch among different security levels, based on the data address expelled by the processor. The security level switching can be achieved by adapting a look up table containing the address with their corresponding security level. In this work we have considered three main security levels they are Confidentiality and integrity, Confidentiality only and no security. The security level classification is done completely based on the user interest and is independent of the operation system or the processor. All the parameter values assigned in the security core are present in the encrypted data and can be accessed in future. If the security level has to be configured, then a fresh memory security protocol has to be designed.

In our proposed implementation the hardware security core links the gap between the processor's cache memory and the external memory's cache memory. All the data transmission between the processor and external memory takes place via, the proposed hardware security core architecture for ensuring the data security. The top module, the hardware security core includes sub modules Encapsulation and De-encapsulation unit for formatting the data, address and the segment id into plain text, a Time stamp generator for generating a count value at that instance and used here as a key for ECC cryptoprocessor, a segment id generator for generating a security level based tag for identifying the security level needed for the data, an internal memory is used here for storing the segment id, TS value along with the corresponding address value and the ECC engine for encrypting the encapsulated plain text into cipher text and to decrypt the stored data in the external memory to the original plain text.

In the Figure 4, when a write instruction is activated, the write line value becomes *1*, thereby enabling the segment id generator. The segment id generator, based on the security level for the input address controls other sub modules.

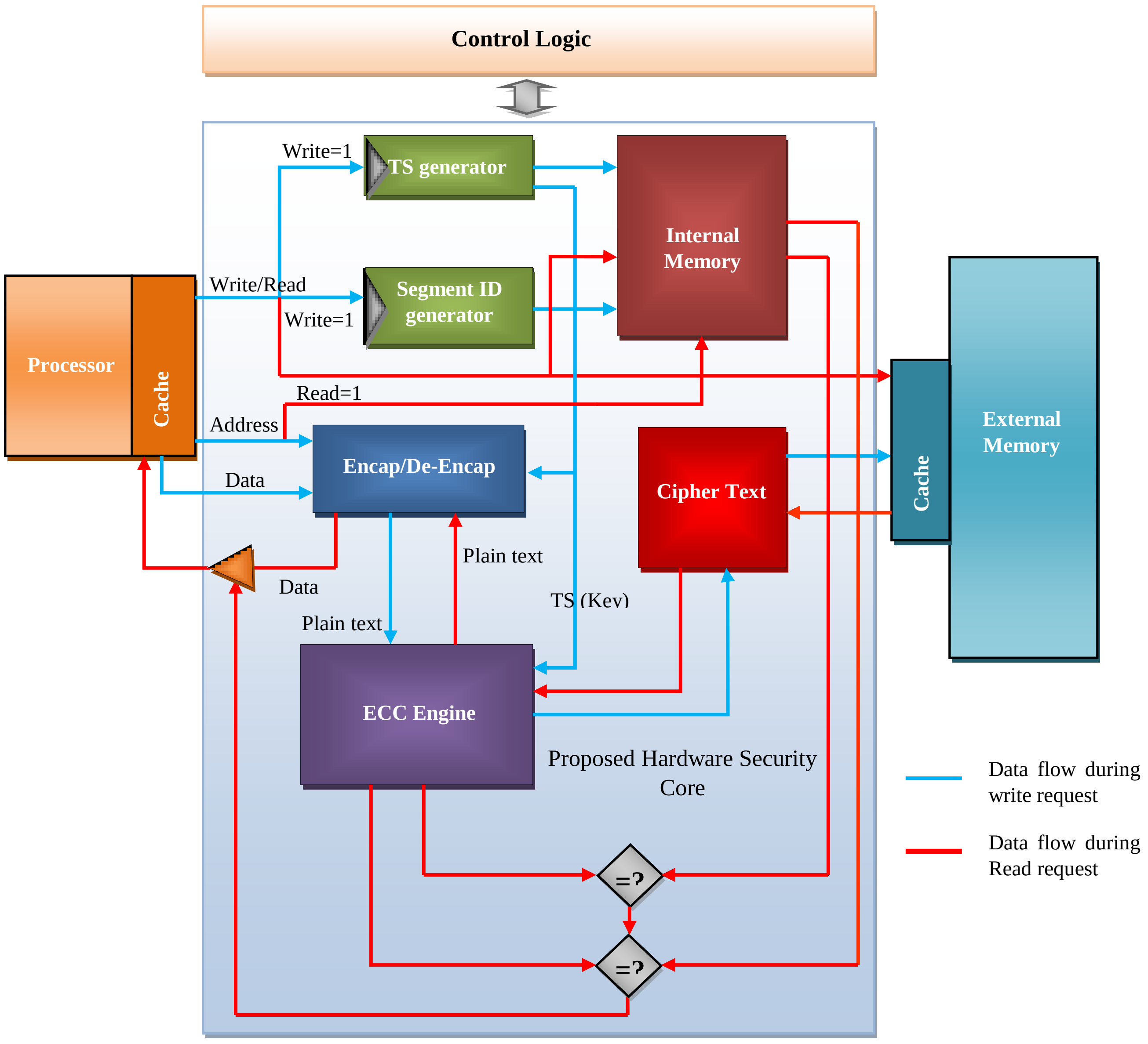

Figure 4. Architecture of Proposed security Core

If the security level for the corresponding address is high then the time scale value at that instant is fed as the private key to the ECC engine and the data, address and the corresponding *TS* value are subjected to the encapsulation process. The encapsulated data is then fed as a plain text *M* to the ECC engine for encryption. The address, segment id, *TS* value are all stored in the internal memory for the integrity checking. If the security level for the address location, where the data has to be written is low, then no security has to be provided for the data corresponding to that address and hence the data can be directly stored in the external memory. The algorithmic representation when the write signal is activated by the processor is shown in Algorithm 1.

Whenever the read instruction is activated by the processor, based on the address location the cipher text stored in the external memory is retrieved and in the meanwhile the security level for that address is identified from the internal memory and by using the TS value (*d*) stored corresponding to the given address, the cipher text is decrypted to a plain text *M*. The plain text *M* is then de-encapsulated to the original format and then the address; *TS* are compared with the stored values for identifying attacks. If the values match, then the de-encapsulated data is allowed to pass to the processor. Algorithm 2, gives a clear idea about the proposed concept for memory data security when the read signal is activated by the processor.

```
Algorithm :1
Input   : Data, Address, write
Output : Ciphertext
while
write = 1
if
seg id = 1 or 2
        then
          TS generator        ⇒ TS + 1
          Plaintext           ⇐ Encap{Address; Data; TS}
          Internal memory     ⇐ Address; Data; Seg id; TS
          Cipher text         ⇐ ECC{Plaintext, TS}
          External memory     ⇐ Cipher text
else
          External memory     ⇐ Data
end if
end
return (0)
```

```
$\text{Algorithm}: 2$
$\boldsymbol{Input} \quad : \boldsymbol{Ciphertext}$
$\boldsymbol{Output} : \boldsymbol{Data, Address, TS}$
$\boldsymbol{while}$
$\boldsymbol{Read} = 1$
$\boldsymbol{if}$
$\boldsymbol{seg\ id} = 1(\boldsymbol{in\,internal\ memory})$
         $\boldsymbol{then}$
             $\boldsymbol{Cipher\ text} \Leftarrow \boldsymbol{External\ memory}$
             $\boldsymbol{Plaintext} \Leftarrow \boldsymbol{ECC}\{\boldsymbol{Cipher, TS}\}$
            $\boldsymbol{De - Encap(Plaintext)} \Rightarrow \{\boldsymbol{Address}; \boldsymbol{Data}; \boldsymbol{TS}\}$
 $\boldsymbol{Integrity\ checking}: \boldsymbol{Internal\ memory}\{\boldsymbol{Address, TS}\} = \{\boldsymbol{Address, TS}\}$
     $\boldsymbol{if}$
                  $\boldsymbol{match} = 1$
              $\boldsymbol{then}$
               $\boldsymbol{Data} \Rightarrow \boldsymbol{Cache\ memory}$
              $\boldsymbol{else}$
              $\boldsymbol{Attack} = 1$
   $\boldsymbol{end\ if}$
$\boldsymbol{ifelse}$
$\boldsymbol{seg\ id} = 2(\boldsymbol{in\,internal\ memory})$
         $\boldsymbol{then}$
             $\boldsymbol{Cipher\ text} \Leftarrow \boldsymbol{External\ memory}$
             $\boldsymbol{Plaintext} \Leftarrow \boldsymbol{ECC}\{\boldsymbol{Cipher, TS}\}$
            $\boldsymbol{De - Encap(Plaintext)} \Rightarrow \{\boldsymbol{Address}; \boldsymbol{Data}; \boldsymbol{TS}\}$
             $\boldsymbol{Data} \Rightarrow \boldsymbol{Cache\ memory}$
              $\boldsymbol{Attack} = 0$
$\boldsymbol{else}$
             $\boldsymbol{Data} \Rightarrow \boldsymbol{Cache\ memory}$
              $\boldsymbol{Attack} = 0$
$\boldsymbol{end\ if}$
$\boldsymbol{end}$
$\boldsymbol{return}(0)$
```

A) Encapsulation/De-Encapsulation

The initial step in our technique for memory security is the wrapping of the address, data and segment id into a single data with a specific format as a plain text for the ECC engine. Then the data, address and the segment id from the cipher text stored in the external memory are unwrapped and separated as the original. This process is handled by a module Encap/ De-encap shown in the Figure 5. The process of encapsulation and de-encapsulation used in our case is depicted in Figure 5.

For example if a data of length 64-bit and the length of the address, TS is 16-bits, 8-bit respectively, then the length of the encapsulated data is with a length of 88 bits, with a format as shown in Figure 5.

B) Time stamp generator

To prevent replay attacks, a simple time stamp generator, such as a counter, is used. As shown in Algorithm 1, the TS value associated with each data address is incremented by 1 after each write to the memory. For each new processor write request, the system will generate a different key since the value of TS is incremented. During a read, the original TS value stored in the internal memory is used for comparative purposes. The retrieved TS value is provided to ECC during the read request. The ECC result will give the same key as the one produced for the write request and the encrypted data will become plaintext.

Read-only data, such as processor instructions, do not require protection from replay attacks because these data are never modified. No TS values are needed for these data so the amount of on-chip TS memory space can be reduced accordingly. Read-only data may be the target of relocation attacks but the address used to compute the ECC guarantees protection against these attacks. The use of time stamps and data addresses for ECC protects read/write data against replay and relocation attacks. If a data value is replayed, the TS used for ciphering will differ from the one used for deciphering. If a data value is relocated, its address will differ from the one used to generate the key. In both cases, the deciphered data will be invalid.

C) Segment id generator

The segment id generator is a simple LUT based architecture used for identifying the security level for a particular address location. When the write instruction is activated, the initial process that takes place is the segment id generation process. The memory is segmented and security level for each address location and their corresponding security level are stored in the LUT as shown in figure. When the write signal is activated by the processor, the LUT is searched for the security level related to that address and with respect to that address the remaining modules are enabled. The three levels of security and their corresponding Segment id are tabulated in table. Same segment ID is used for all address belonging to a particular application segment.

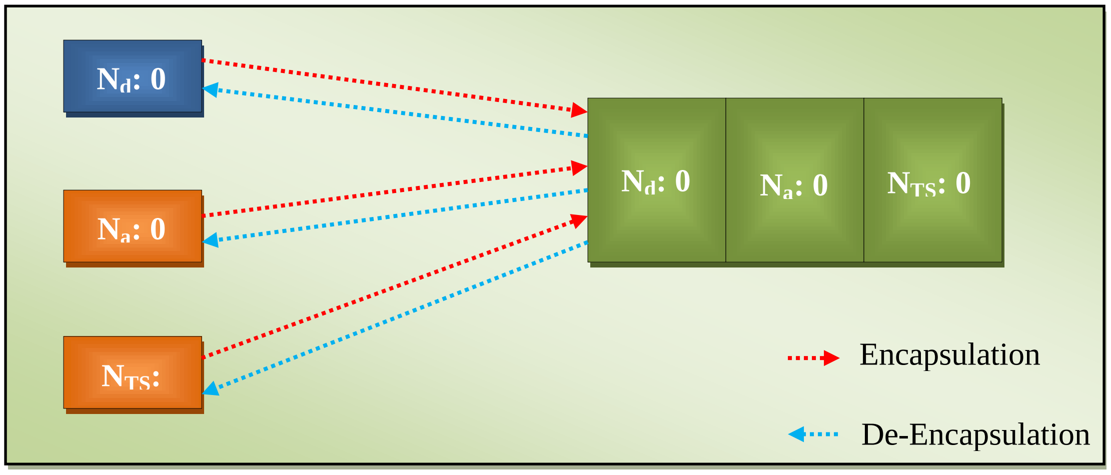

Figure 5: Encapsulation/ De-encapsulation module

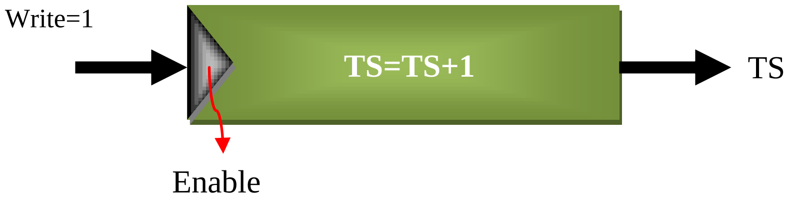

Figure 6: Time stamp generator

Table 1: Security level and their corresponding segment id

Security Level	Segment Id
No security	00
Confidentiality only	10
Confidentiality and Integrity	11

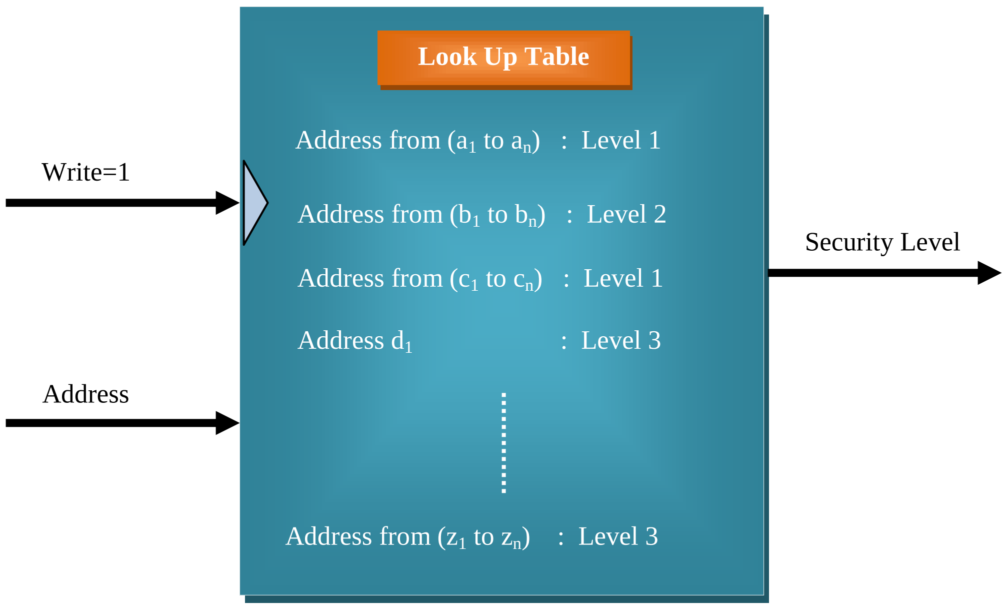
Look Up Table
Address from (a_1 to a_n) : Level 1
Address from (b_1 to b_n) : Level 2
Address from (c_1 to c_n) : Level 1
Address d_1 : Level 3
Address from (z_1 to z_n) : Level 3
Write=1
Address
Security Level

Figure 7: LUT based segment id generator

If the security level for the input address belongs to the segment which needs no security then the LUT output will be '00', thereby disabling the encap/De-Encap, TS generator and the ECC engine. If the address represents a segment for which there should be a need for confidential checking, then the segment id generated by the generator will be '10'. This segment id enables all the sub modules, leaving only the storing of data in the internal memory, since the data corresponding to the given address does not need an integrity checking. In the case of the segment that needs confidentiality and integrity, the segment id is '11' and this enables the ECC engine, encap/De-Encap and the TS generator.

D) ECC Engine

The ECC engine here provides security storage by encrypting and decrypting the data. The modified form of the ECC crypto processor using double point multiplication is adapted in our work. The working process of the ECC engine can be discussed below. Initially, the public key is generated whenever write signal goes high and let us consider the computed public key as K_{pu}. Since the generation of the public key does not depend only on the write signal of the processor output, this is similar as that for the conventional method. The public key generation is followed by an encryption process.

Encryption

For the encryption process, the plain text *M*, expelled from the encapsulation module is fed to the ECC engine along with the *TS* value. The TS generator designed should be of a LFSR type, since the private key to the ECC is a random number ranging from a value *1* to *n-1*, where *n* is the used defined bit length. Two random points P_1 and P_2 that lies in the considered elliptic curve equation and two other random numbers r_1 and r_2 are also defined from the range *1* to *n-1*. By using the double point multiplication, r_1 P_1 and r_2 P_2 are computed. The cipher text composed of two parts. The first part C_1 is obtained by performing point addition between the two points P_1 and P_2 and then performing finite field multiplication between the added point and the *TS* value at that instant. The second part of the cipher text is obtained by adding the plain text *M* , with the point which is obtained as a rest of point addition between the two points r_1 P_1 and r_2 P_2, that are generated in the initial public key generation stage and multiplying it with the obtained *TS* value. Both the cipher text C_1 and C_2 are merged and then stored in the external memory. The modified encryption algorithm is shown in Algorithm 3.

```
Algorithm : 3
Input   : $P_1, P_2, M, r_1P_1, r_2P_2 \in E$ and $TS \in \{1-(n-1)\}$
Output : $C_1, C_2 \in E$
1. Load TS
2. $C_1 = TS \times (P_1 + P_2)$
3. $temp = TS \times (r_1P_1 + r_2P_2)$
4. $C_2 = M + temp$
5. return $(C_1, C_2)$
```

Decryption

During the decryption process of the ECC engine, the merged cipher texts C_1 and C_2 and the private key *TS* stored in the internal memory for the corresponding read address executed by the processor are used to decrypt the original plaintext from the encrypted cipher texts stored in the external memory. The decryption process of our ECC engine can be better understood from the algorithm given below.

```
Algorithm : 4
Input   : $C_1, C_2 \in E$ and $r_1, r_2 \in \{1-(n-1)\}$
Output : $M$
2. $M = C_2 - \{TS \times (r_1 + r_2)\}C_1$
3. return $M$
```

The plain text thus obtained is then de-encapsulated and then based on the segment id, the integrity checking takes place. (See algorithm 2).

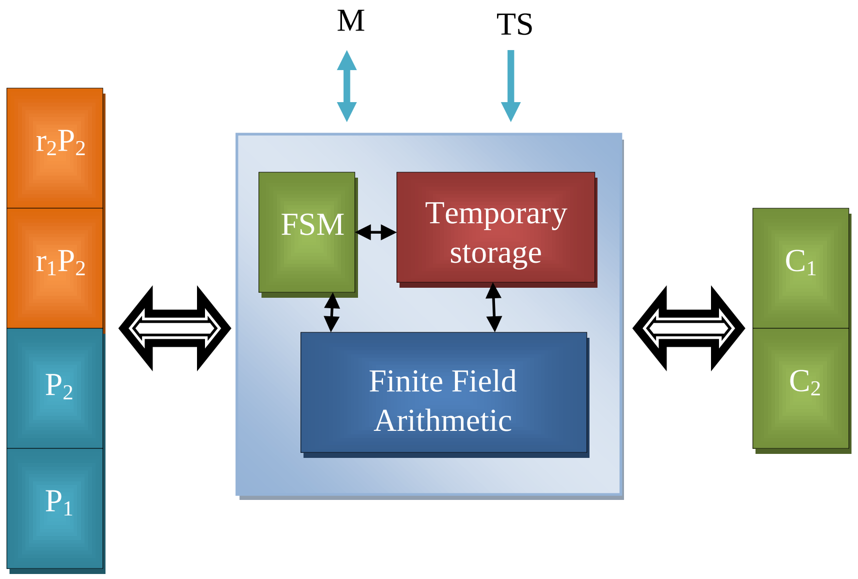

Figure 8: ECC Engine

The ECC engine shown in Figure 8 operates based on the encryption and decryption process discussed in algorithms. The FSM controls the working of the complete ECC module. The finite field arithmetic incorporates the point addition, dual point multiplication, squarer and the finite field inverter. The operation of the ECC engine is similar as given in algorithms 3 and 4.

V. RESULT AND DISCUSSION

In order to validate the benefits of proposed approach, hardware security core is implemented along with a DWT (Discrete Wavelet Transform) processor and associated memory. The complete architecture discussed in the proposed method is synthesized using Xilinx ISE 14.1 Virtex-4 xc4vlx200-11-ff1513 as a target FPGA device. The RTL schematic and the simulation of proposed architecture is given in Figure 9 and Figure 10 respectively.

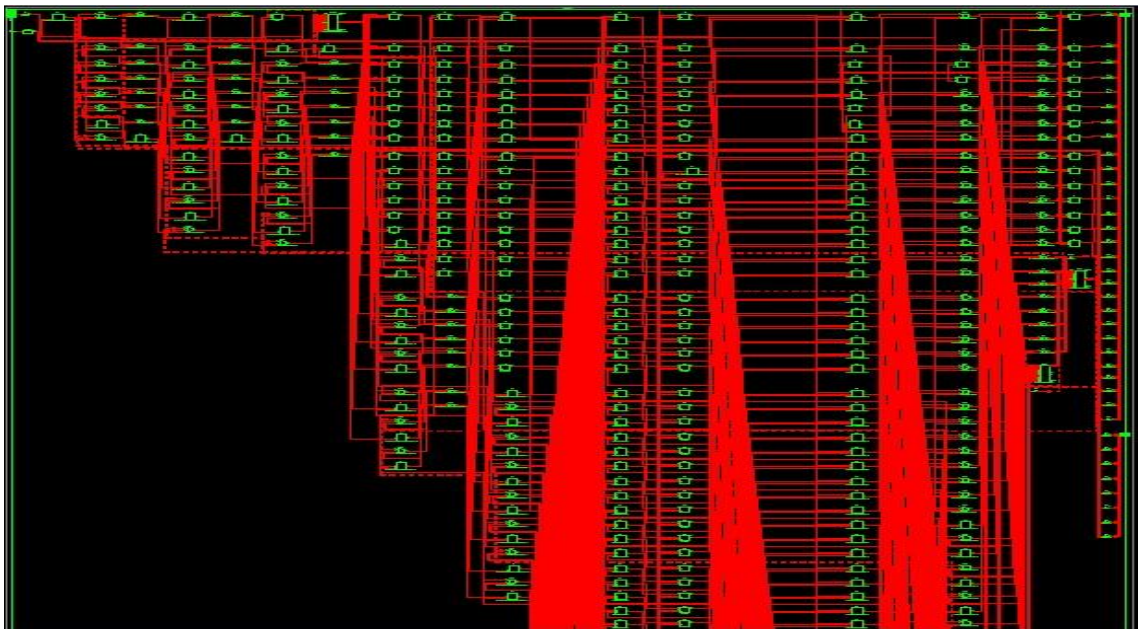

Figure 9: RTL schematic of proposed hardware security core

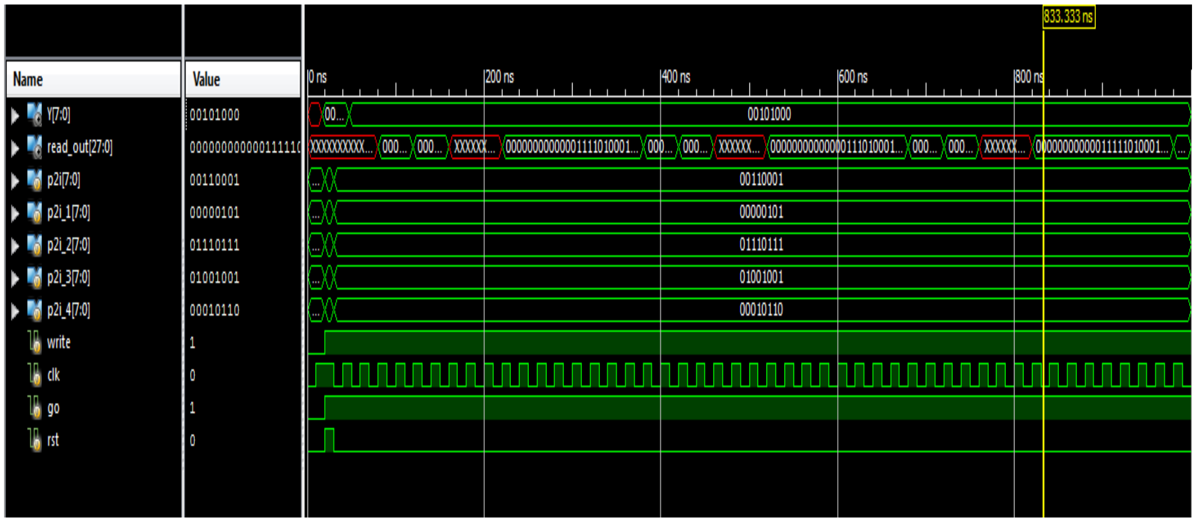

Figure 10: Simulation of proposed hardware security core

Table 2: Comparison of DWT and proposed hardware security core incorporated DWT modules

Parameters	Only DWT	DWT+Proposed Hardware security core	Overhead %
Target Device	xc4vlx200-11-ff1513	xc4vlx200-11-ff1513	-
Slices	33	163	-79%
4 input LUTs	58	250	-76%
DSP48s	3	3	0%
GCLKs	1	1	0%
Delay	4.615ns	4.630ns	-0.3 %(~0%)
memory usage	292364 Kb	391064 Kb	-25%
Maximum Frequency	-	432.302MHz	-

Table 3: Power comparison for DWT and proposed hardware security core incorporated DWT modules

Frequency(MHz)	Only DWT			DWT+Proposed Hardware security core		
	Dynamic (mW)	Static (mW)	Total (mW)	Dynamic (mW)	Static (mW)	Total (mW)
100	9	1344	1353	37	1344	1384
200	16	1345	1361	41	1347	1389
300	20	1345	1365	46	1348	1393
400	23	1346	1369	50	1348	1398

Comparing with the DWT architecture without security core, which is implemented in the same target device, our architecture with security core has an area overhead of 79% in terms of slices and 76% in terms of LUTs. This overhead is due to the additional hardware resources needed for providing security to the processor architecture. Both the architectures include only 3 DSP48s and 1 global clock. The delay time exhibited by our proposed architecture is 0.3% is greater than that of the conventional DWT architecture without the security module. Since the overhead reported here is a negligible value the delay overhead will not be considered and hence our security core incorporated DWT processor will process the data with the same delay time as that is reported by the conventional DWT processor. The memory usage of the proposed module also has a memory overhead of about 25%, which is because of the additional internal memory we have adapted for storing the address, segment id and the TS value. The hardware security core also reported an overall frequency rate of 432.302MHz.

The bar chart shown in Figure 11 depicts the difference in power consumption reported by normal DWT and DWT processor architecture with the proposed hardware security core. Comparing the power consumption, the static power remains the same, whereas the dynamic power differs a negligible amount thereby contributing the slight increase in the overall power consumption by the complete module. The area and latency reported by various levels of security is tabulated in Table 3 below.

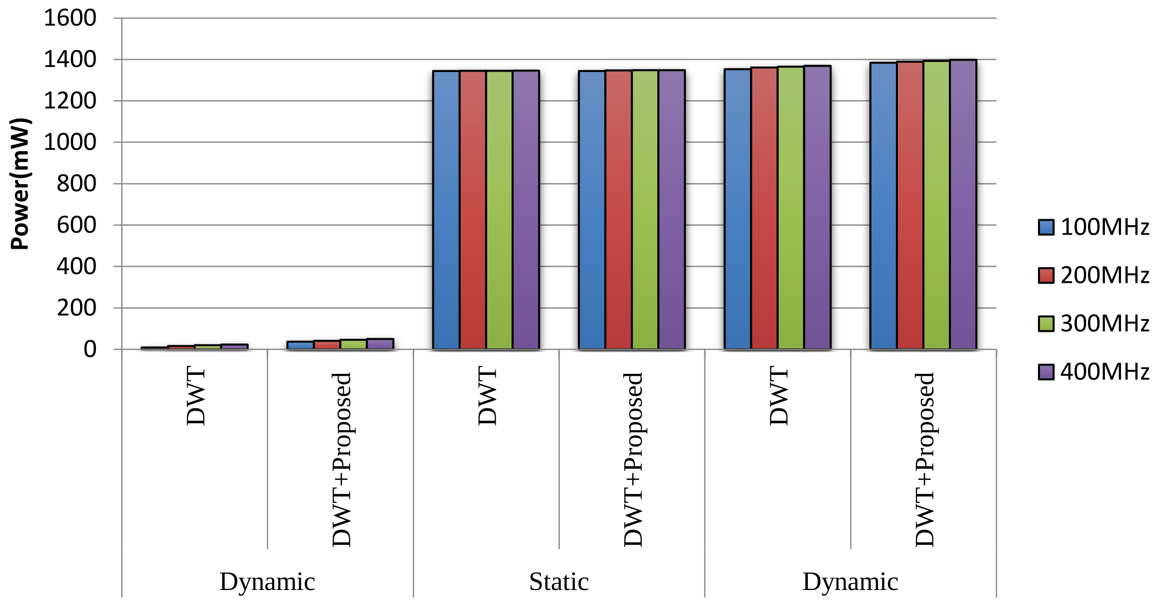

Figure 11: Power comparison with respect to frequency

Table 3: Area and Latency reported by various levels of security

Level of Security	Latency Time	Resources Utilized
No Security	4.615ns	44 slices,20Flipflops,65 LUTS
Confidentiality	4.615ns	48 slices,29Flipflops,68 LUTS
Confidentiality and Integrity	4.630ns	69 slices,65Flipflops,77 LUTS

VI. CONCLUSION

In this paper, a configurable architecture for hardware security core offering security services to off chip memory in embedded system has been proposed. Core can be configured to offer three different levels of security to data. Security core houses confidentiality and Integrity engine based on area efficient ECC algorithm. The proposed ECC based solution for memory security shows no latency delay even after invoking different and stringent security levels. The experiment was conduction by connecting the designed security core with a re-designed DWT processor for evaluating the area, power and performance of the implemented design. Experimental results shows that the proposed design even though have slight overhead in terms of area, the latency reported for the design and power consumption is almost stable for three levels of security. This security core is not adversely affecting the processing speed and not contributing to power consumption.

REFERENCES

[1]. Trimberger, S.M and Moore, J.J, "FPGA Security: Motivations, Features, and Applications", Proceedings of the IEEE, Vol.2, No. 8, August 2014.

[2]. Apostolos P. Fournaris and Nicolas Sklavos, "Secure embedded system hardware design – A flexible security and trust enhanced approach", Elsevier Journal of Computers and Electrical Engineering Volume 40, Issue 1, January 2014, Pages 121–133.

[3]. Jakub Szefer, "Towards fast hardware memory integrity checking with skewed Merkle trees", Proceeding HASP '14 Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy,Article No. 9.

[4]. R. Vaslin, G. G. ans Jean-Philippe Diguet, E. Wanderley , R. Tessier, and W. Burleson. "Low latency solution for confidentiality and integrity checking in embedded systems

with off-chip memory", Workshop on Reconfigurable Communication-centric Systems-on-Chip, June 2007.

[5]. Sunil Devidas Bobade and Dr. Vijay R .Mankar, "VLSI Architecture for an Area Efficient Elliptic Curve Cryptographic Processor for Embedded Systems", Proceedings of IEEE International Conference on Industrial Instrumentation and Control (ICIC 2015), College of Engineering, Pune , pp: 1038-1042,May 2015.

[6]. Sunil Devidas Bobade and Dr. Vijay R. Mankar, "Area Efficient Implementation of Elliptic Curve Point Multiplication Algorithm", International Journal of Advanced Computer Science and Applications (IJACSA), United Kingdom, ISSN: 2156-5570, Vol. 6, Issue 04, pp: 24-34, April 2015.

[7]. Amir Moradi, Alessandro Barenghi, Christof Paar and Timo Kasper, "On the Vulnerability of FPGA Bitstream Encryption against Power Analysis Attacks," Proceedings of the 18th ACM conference on Computer and communications, pp: 111-124, October 2011.

[8]. A.Kaleel Rahuman, Dr. G.Athisha, "Reconfigurable Architecture for Elliptic Curve Cryptography," Proceedings of the International Conference on Communication and Computational Intelligence, pp.461-466, December- 2010.

[9]. Reza Azarderakhsh and Koray Karabina,"A new double point multiplication algorithm and its application to binary elliptic curves with endomorphisms", IEEE Transactions on Computers, No.99, May 2013.

[10]. Satyajeet Nimgaonkar, Mahadevan Gomathisankaran , Saraju P. Mohanty, "TSV: A novel energy efficient Memory Integrity Verification scheme for embedded systems", Journal of Systems Architecture, No. 59 , pp 400–411,2013.

[11]. Jeremie Crenne, Romain Vaslin, Guy Gogniat, Jean-Philippe Diguet, Russell Tessier and Deepak Unnikrishnan, "Configurable memory security in embedded systems", ACM Transactions on Embedded Computing Systems (TECS), Vol.12, No.3, March 2013.

[12]. Ma Haifeng, Yao Nianmin, Cai Shaobin and Han Qilong, "Memory Confidentiality and Integrity Protection Method Based on Variable Length Counter", Journal of Algorithms & Computational Technology, Vol.8, No.4, 2014.

[13]. Satyajeet Nimgaonkar, Mahadevan Gomathisankaran and Saraju P. Mohanty, "MEM-DnP—A Novel Energy Efficient Approach for Memory Integrity Detection and Protection in Embedded Systems" Circuits, Systems and Signal Processing, Vol. 32, No. 6 , pp 2581-2604,2013.

[14]. Bao Liu and Brandon Wang, "Reconfiguration-Based VLSI Design for Security", IEEE Journal on Emerging and Selected Topics In Circuits And Systems, Vol.5, No.1, March 2015.

[15]. Apostolos P. Fournaris and Nicolas Sklavos, "Secure embedded system hardware design – A flexible security and trust enhanced approach", Elsevier Journal of Computers and Electrical Engineering Volume 40, Issue 1, January 2014, Pages 121–133.

[16]. Zhenglin Liu, Qingchun Zhu, Dongfang Li, and Xuecheng Zou, "Off-Chip Memory Encryption and Integrity Protection Based on AES-GCM in Embedded Systems", IEEE Design & Test, Vol.30, No.5, pp: 54 - 62, 2013.

Sunil Devidas Bobade obtained his Engineering Graduate Degree in Electronics and Telecommunication Engineering from VYWS College of Engineering, Amravati (India) in 1994, Post Graduate Degree in Electronics Engineering from S.G.B.Amravati University, Amravati (India) in 2002.

He has been in the field of teaching since last 19 years and is presently, working as an Assistant Professor in Department of Information Technology, Datta Meghe College of Engineering, Navi Mumbai. He is also working as a research scholar in S.G.B.Amravati University and is working on development of area efficient algorithms for protection of memory of embedded systems.

Dr. Prof. Vijay R. Mankar received his B.E. degree in Electronics and Power Engineering from Government College of Engineering, Amravati (India) in 1986, M.Tech. in Electronics Engineering from erstwhile Visvesvaraya Regional College of Engineering, Nagpur (India) in 1990 and Ph.D. from S.G.B. Amravati University, Amravati (India) in 2009.

He has been in the field of teaching since last 23 years and is presently, working as a Deputy Secretary, MSBTE, Pune Region. He has served as Head of Department and the Professor, with Department of Electronics Engineering, Govt. Polytechnic, Amravati. He has been active in the research as well. His research interests are in Neural Networks, Design of Embedded system, Data security. He has published over twenty five research papers in journals and conferences of national and international repute.