ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Implementation of Dedicated Hardware Using Encryption Technology

Gaurav Berad[1], Prof. Mayur C Akewar[2]

ME Student, Dept. of Computer Engineering, RMD, Sinhgad School of Engineering, Pune, India

Assistant Professor, Dept. of Computer Engineering, RMD, Sinhgad School of Engineering Pune, India

ABSTRACT: In this Cyber-Age where every major task, action, or operation executed in real life can be similarly executed in Virtual (E)/ Digital/ Second Life with more ease, be it banking, commerce, work, or communication; digitization has provided seamless options to operate, connect and share. Such a vast pool of ease and liberty gives birth to lack of security and added risk of losing sensitive data and IPR. There are established methods and security solutions which claim to be successful to a certain extent. And in cases where such software based or limited hardware based solutions are not satisfactory there arieses the need of a more developed and mature security solution. Dongles for hardware protection have been present for a while now, but unfortunately, most of these solutions were only effective for a short period, until these methods were circumvented and time has showed that solutions that offer full protection from this phenomenon are impossible. A hardware protection scheme based on dongles provides a Highly Secure Protection scheme. An AES algorithm is implemented on FPGA stage to improve the security of data in transmission. AES algorithms can be implemented on FPGA with a particular deciding objective to speed data taking care of and reduce time for key generation.

KEYWORDS: Advanced Encryption Standard (AES), Field-Programmable Gate Array (FPGA), Human Interface Device (HID), Encryptor, Decryptor.

I. INTRODUCTION

Data is a vital asset. A huge segment of authoritative spending plan is spent on overseeing data. So keep that information safe and secured. Subsequently either programming or hardware security must be suited that information. Writing computer programs is a general term used to depict a gathering of PC tasks, methods, and documentation that perform some errand on a PC system. Hardware is best depicted as a device that is physically joined with something or something that can be physically touched. As hardware can't be hacked or balanced it is more secure than programming. There are many ways to protect information from misuse; choices can be summed up into two categories as:

Hardware-base protection: A hardware device is integrated with software and used to protect information and intellectual property.

Software-base security: A software license key, is input or generated by user's computer to protect information and intellectual property.

Equipment base assurances utilize a physical gadget, regularly known as dongle, USB equipment key or security key. To actualize, the dongles firmware is coordinated with the software. After which programming just runs if dongle is physically present on the PC or machine. Furthermore, dongle controls how end client utilizes the software. Whether it's opportunity or utilizes based limitations or restricting the modules and highlights accessible, the end client's entrance is overseen and upheld by the dongle.

AES can be executed in hardware or software but, hardware usage is more suitable for rapid applications progressively. Essential target AES equipment use is high throughput plan and low-region work. The last gives most endeavors to minimize size of the outline and bring down the force utilization [5].

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

II. RELATED WORK

Living in an information age, it is noticeable how much the computers are used and how important they are for many of our everyday tasks. Software solutions exist for solving many problems we encounter. Developing a specific software usually takes a lot of efforts and investment. The written lines of source code have a high production value, but no real material value. Therefore, it seems impossible to create a security mechanism that would be able to resist forever. A practical solution to this challenge is to create a security system that is not feasible to attack. In the best case, the cost to circumvent the software protection should be more expensive than the price of the software itself and the effort would be as high as recreating the software. Software protection dongles, as one of the more secure alternatives, are being used for some years now. Therefore, we aim at creating a new design for the dongle protected software, which will provide a much higher level of binding (linkage) between the dongle and the software.

Code obfuscation speaks to an entire arrangement of programming assurance systems that increase present expectations for the assailant as far as abilities or assets. They are reasonable to send contrasted with equipment helped arrangements, as no extra equipment should be gained. They are adaptable as strategies can be connected to a specific degree that meets security versus execution exchange offs set by the product proprietor. Besides, obfuscation procedures can be effectively incorporated amid programming advancement, as the strategies are programming as it were [7].

Hardware base security:

The techniques for software base security have appeared to have powerless security highlights, as they were broken at some point or another. Hence, a ton of endeavors are being put on an option measure - the utilization of hardware based arrangements - dongles. An expanding number of embedded gadgets execute some security usefulness, for occasion, smart cards (keeping money, SIM, open transport, access control, international IDs), auto keys, set-top boxes (pay TV), media players, cellular telephones, tablets, restorative inserts, and so forth. These gadgets use cryptographic algorithms that are secure against numerical cryptanalysis. This implies a framework's security depends on the mystery of an alleged cryptographic key, and that there are no numerical alternate routes that permit breaking the framework. Be that as it may, in the late 90s, the security of such gadgets has been appeared to additionally rely on upon the algorithm usage [2].

III. PROPOSED SYSTEM

As computing frameworks turn out to be more fundamental to our day by day lives, it turns out to be perpetually vital that the administrations they give are accessible at whatever point we require them. We should likewise have the capacity to depend on the respectability of the frameworks, and along these lines the data that they hold and give. Besides, our general public and our economy rely on specific bits of data being held in certainty. We need to be guaranteed that they will work precisely not surprisingly, and that they will continue working even notwithstanding catastrophes, mischances, or planned endeavors to meddle with or keep their capacity.

While hardware might be a wellspring of frailty, for example, with microchip vulnerabilities malignantly presented amid the assembling process, hardware based or helped PC security likewise offers a different option for programming just PC security. Utilizing gadgets and strategies, for example, dongles, trusted stage modules, interruption mindful cases, drive locks, debilitating USB ports, and versatile empowered access might be viewed as more secure because of the physical get to (or advanced indirect access) required with a specific end goal to be traded off [3].

A USB Dongle is a little bit of hardware that interfaces with a portable PC, desktop or server PC. It supports plug and play technology.

Generally USB Dongles have two different technologies to protect software and files. First one is Authentication technology, the keys stored into the dongle is used to run application if desired key is found. Second one is code porting innovation; a part of the code is ported inside the dongle. In the proposed system agents Key Pair is stored in the USB Dongle in the form of encoded manner and Crypto-Agent is ported into the dongle.

It has some important features like File Protection System, Code Port solution, Smart Technology and Automatic Self-locking Mechanism, Global Unique Serial Number and Built-in Timer.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

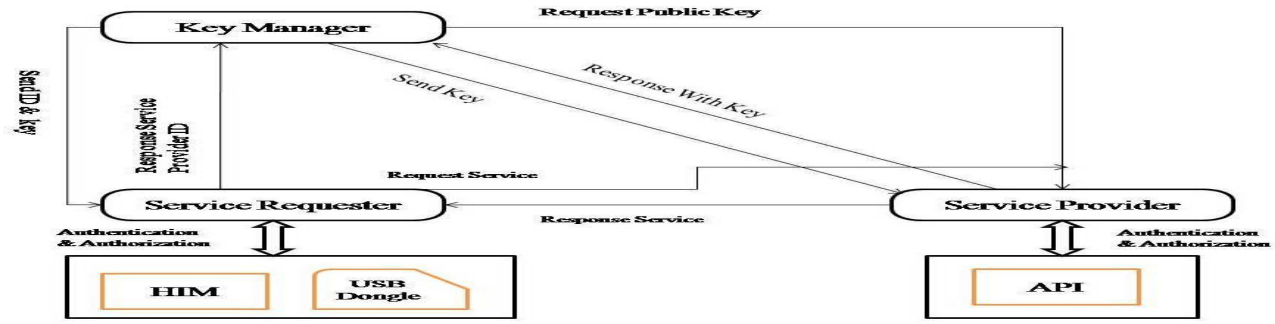

Fig 1: Proposed System

A. *Design Work flow:*

The Figure 1 shows architecture of the proposed system, which consists of server and client environment. The server environment consists of Key Manager (KM), Platform Managers and Service Providers (SPAs – Service Providing Agent). Client environment consist of Service Requestors (SRAs – Service Requesting Agent).Each SPA and SRA has a USB Dongle which is configured uniquely for the purpose of identifying a distinct and trusted agent in system[13].

B.Description of Algorithm:

AES was institutionalized by National Institute of Standards and Technology (NIST) in 2001 got to be Federal Information Processing Standard FIPS-197.Where Rijndael algorithm by Joan Daeman and Vicent Rijimen was chosen as standard AES algorithm. The AES is private or symmetric block cipher which utilizes the same key for encryption and decoding is more suitable for quicker execution. The AES is a symmetric key for both encryption and decryption. AES algorithm equipped for encoding and decoding piece size 128 piece information utilizing figure keys of 128, 196 or 256 bits (AES128, AES196 and AES256).

AES can be executed in hardware or software but, hardware usage is more suitable for rapid applications progressively. Fundamental objective AES hardware execution is high throughput outline and low-region plan work at most elevated working recurrence.

AES algorithm is symmetric block cipher that procedures the state displaying from 128 bits information piece utilizing a key of 128, 192 or 256 bits length over and again. In encryption process round capacity comprises of four distinct changes SubBytes, ShiftRows, MixColumns and AddRoundKey however last round capacity without MixColumns changes. A ShiftRows changes are genuinely reliant on state-wise operation of cyclic line moving and MixColumns select 4 byte section operation done at all the while. Similarly chipper decryption round capacity comprises of four unique changes InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey, yet last round capacity without InvMixColumns changes[13].

IV. IMPLEMENTATION RESULTS AND COMPARISON

Following are general steps to be followed for the implementation as:

Step 1: Initialize Dongle and API.

Step 2: Establish Secure Connection.

Step 3: Read Data from Main Memory from the buffer of the target application.

Step 4: Start Communication with the of AES.

Step 5: using Sbox SubBytes(). It provides invertible transformation of segments.

Step 6: ShiftRows(). Applied for shifting of rows. amount of shifting shift(r; N_b)

$S'_{r,c}=s_r;(c+shift(r;N_b))$ mod N_b

Step 7: MixColumns(). Usig the some specified byte length. Transformation s'(x) is given by the multiplication of the input columns(x)with the polynomial a(x) and reduced modulo (x4 + 1):

$S'(x) = a(x) \neq s(x) \bmod (x4 + 1)$

For(upto byte length)

{

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Mix the colums

}

Step 8: AddRoundKey(). Portion of key 'w' is is used for N-1 times. Key is uses up 4*Nb = 16 bytes. at each round, i.e. for 1< round < Nr , a different 32-bit Round Key w_i is added:

$[s'_{0;c}; s'_{1;c}; s'_{2;c}; s'_{3;c}] = [s'_{0;c}; s_{1;c}; s_{2;c}; s_{3;c}] \oplus [\omega_{(round * Nb)+c}]$ where $0 < c < N_b$

. For(upto length of expanded key)

{

Apply key for each block

}

Step 9: For the decryption of data step 4 to step 7 are applied in inverse order.

Step 10: Write the data into application buffer in the main memory.

Step 11: Close/Exit Encryption when dongle removed or application instance from launcher is ended.

We utilize two arrangements of state registers, each comprising of sixteen 16-bit registers, relating to the two shares of the state. The MixColumns and the Key XOR operations are likewise performed with two shares, as the key and the state registers are 256 bits suggesting the two shares [1].

This usage of the S-box requires four info offers. We share the key in two shares and XOR them with two of the plaintext offers before the S-box operation. Other than the mutual info, the S-box needs 20-bits of arbitrariness *r*. The initial two yield offers $sbout_{1,2}$ are composed to the state register S_{33} though the remaining share $sbout_3$ is composed to enroll P_3. The information in the state registers are moved to one side for the accompanying 16 cycles so that the following yield of the S-box can be put away in the same registers. Amid this move, the information in P_3 is XORed with the second share of the S-box yield, which is in the state register S_{33}, to lessen the quantity of shares from three to two.

The ShiftRows operation is performed in the nineteenth clock cycle with an unpredictable level movement. In the accompanying four clockcycles, the data in the registers S_{00}, S_{10}, S_{20}, and S_{30} are sent to the MixColumns operation, whatever remains of the registers are moved to one side on a level plane and the yield of the MixColumns operation is composed to the registers S_{03}, S_{13}, S_{23}, and S_{33}. The MixColumns operation is executed section astute as in [12] and with two shares working in parallel.

In the accompanying AES rounds, we build the quantity of shares of the S-box data from two to four, utilizing 24 bits of arbitrariness, one clock cycle before the S-box operation. To accomplish this sign, sig1 is dynamic for 16 clock cycles, beginning from the last clock-cycle of each round. We isolate the expansion of the quantity of shares and the nonlinear operation with registers to accomplish the non culmination property. The two extra shares are put away in P_0. The two shares in S_{00} are XORed with the two shares of the relating round key byte and sent to the S-box together with the two shares in P_0.

In order to realize high-speed processing and area reduction, this study introduces arithmetic processes suitable for hardware during encryption and decryption. First, a shift process used for encryption is replaced by a bit selection process. Because of this substitution, the shift process can be realized in wiring. The mixture process usually requires t x 3 = 78 calculations. The proposed architecture divides the processing by inserting a register between calculations and introducing a loop process, which reduce the required calculations to 26. Dividing the processing of each round enables high-speed processing, Cryptographs round processing enciphers a 64-bit plaintext. However, the actual processing is performed every 32 bytes. Therefore, the round is divided into the left and right parts to perform left and right processing with two clock signals, which raises the operating frequency and improves latency.

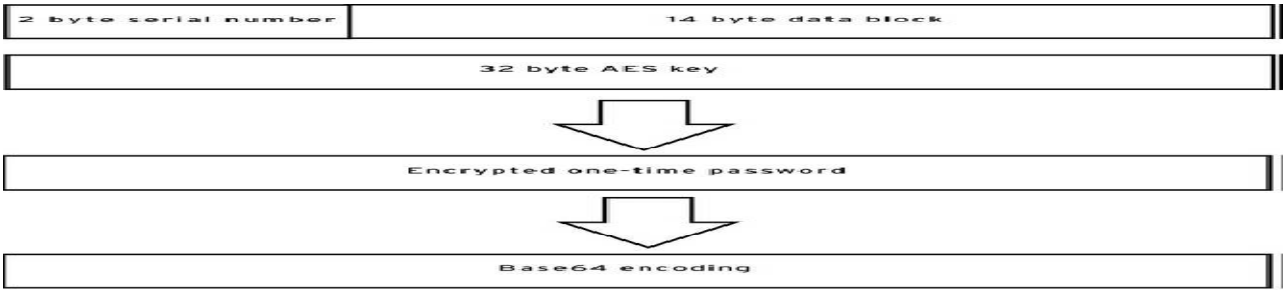

Fig. 4. Implementation of AES Based Encryption in Dongles.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

We implemented an FPGA design that efficiently performs AES video encryption scheme. The described circuits have been implemented in VHDL using the Model Technology's ModelSim Simulator and synthesized, placed, and routed using target device of Xilinx (Xilinx Spartan 3A XC3SD3400A-4FGG676C FPGA). The architecture was simulated for verification of the correct functionality, by using the test vectors provided by the AES standard. Four performance metrics such as the area (slices), the clocking frequency (Mhz), the power consumption and the throughput (Gbps) were computed.

Table 1presents detailed results for this implementation.

XC3SD3400A-4FGG676C	Area (slices)	Frequency (Mhz)	Power (mW)	Throughput(Gbps)
AES-CTR	3047	100.329	4.296	4.28

Table 1 FPGA Synthesis Results

It is apparent in table that AES-CTR processor design provides data throughput advantage. Our circuit has a 3x-unrolled core. It is clear that the critical path inside the core of AES-CTR processor increases with the degree of unrolling. Although the unrolled designs process data in fewer clock cycles than the basic designs, the longer critical path in the unrolled designs means that the maximum clock frequency decreases. Going from a basic quasi-pipelined AES-CTR design to the 3x-unrolled designs reduces the number of clock cycles by a factor of 3. The processor engine was able to operate at 100.329 MHz. In the case, the data is processed at a rate of 4.28 Gbits/sec. So the throughput makes the AES video encryption system feasible for real-time applications. Furthermore, during self-test at 50 mhz, the AES-CTR processor consumed 4,28 mW.

When evaluating a given implementation, the throughput of the implementation and the hardware resources required to achieve this throughput are usually considered the most critical parameters. We synthesized our design using two targets: the Spartan3A DSP 3400-4FGG676C which is the most suitable for our architectures, the Virtex2 XC2V6000 which we used for providing accurate comparisons with existing schemes.

Table 2 compare our implementation with several others works reported in the literature in terms of AES-Encryption only, AES Encryption- Decryption and architecture type . The performance is compared in terms of frequency, the area, the throughput (d) and the number of the block RAM.

References	**Architecture**	**Area (Luts)**	**# of Bram**	**D Gbps**	**TPS**
XC2V1000	Rolling Encryption	1743	-	1.850	0.106
XC2V1000	Rolling Encry/Decry	3555	-	1.049	0.02
XC2V1000 -5	Rolling Encry/Decry	1122	8	1.941	0.17
XC2V6000-6	Pipelinedparallel En	3576	80	24.92	0.69
XC2V6000-6	Pipelined Encry/Decry	17479	18	49.40	0.28
XCV2000E-8	Pipelined Encry/Decry	16693	-	23.65	0.14
XC3S2000-5	Pipelined Encry/Decry	17425	-	25.10	0.14
XC2VP70-7	Pipelined Encry/Decry	7761	400	16.08	0.207

Table 2: Performance comparison results

The introduced system in supports the AES encryption-decryption scheme with rolling architecture. The focus of was to implement a pipelined architecture for AES encryption. presents a pipelined architecture for AES encryption-decryption. Using the throughput metric the proposed design is more proved better . When compared with the results show higher reduction in area and number of BRAMs. To achieve more accurate measure of chip utilization, CLB slice count as chosen as the reliable area measurement. Therefore, to measure the hardware resource cost associated with an implementation's resultant throughput; the throughput per slice (TPS) metric is used. We defined it as

$$TPS = \frac{Throughput}{\#CLB slices\ Used}$$

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2016

Using the TPS metric our design is proved better by compared with Other architectures for high throughput, the results show higher reduction in number of BRAMs. Our design has 6 embedded RAM. The BRAMs are used in the keysRound storage.

V. CONCLUSION AND FUTURE WORK

We presented a novel programming assurance approach that joins solid hardware programming tying. We utilized AES guideline as a part of request to make it suitable for application in programming assurance situations and consolidated it with a control stream chart plan. The AES guideline can perform a full AES encryption (decryption) round without releasing cryptographic keys or halfway consequences of the calculation to an assailant that has full control over the framework the project is running on. As hardware is more secure than the software, these frameworks can give high security to the hacking. The proposed framework is to be executed on FPGA it is conceivable to test the hardware of AES calculation before taking a shot at backend. The main drawback of proposed is framework is no adaptability for determination of Key length. Once the gadget is fabricated nobody can change the key length subsequently it is essential to pick right key length relying on the security required.

So, in this paper we applied the AES algorithm toward the cryptography of data streams. To express the tradeoffs between security and real time application requirements.

REFERENCES

1. Begül Bilgin, Benedikt Gierlichs,Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen, *Senior Member,IEEE" Trade-Offs for Threshold Implementations Illustrated on* AES" IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 34, NO. 7, JULY 2015.
2. P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO* (LNCS 1666). Berlin, Germany: Springer, 1999, pp. 388–397.
3. Jan CAPPAERT "Code Obfuscation Techniques for Software Protection", Dissertation for deg of Doctor in Engineering, Ghent University, April 2012.
4. Kumaravelu R, Kasthuri N, Distribution of Shared Key (Secret Key) using USB Dongle based identity approach for authenticated access in Mobile Agent Security, Internation Conference on Communication and Computational Intelligence (2010), 558-562.
5. Pritamkumar N. Khose, Vrushali G. Raut " Hardware Implementation Of Aes Encryption And Decryption For Low Area & Power Consumption " IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308.
6. Onkar S. Dhede, S. K. Shah, "A Review: Hardware Implementation of AES Using Minimal Resources on FPGA," International Conference on Pervasive Computing (ICPC) 2015.
7. Mansoor Ebrahim, Shujaat Khan, Umer Bin Khalid, "Symmetric Algorithm Survey: A Comparative Analysis", *International Journal of Computer Applications*, Vol. 61 No. 20, pp. 12-19.
8. Sebastian Schrittwieser, Stefan Katzenbeisser, Georg Merzdovnik, Peter Kieseberg, Edgar Weippl, "AES-SEC: Improving software obfuscation through hardware-assistance" 9th International Conference on Availability, Reliability and Security 2014.
9. Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in Cryptology—CRYPTO 2001*, pages 1–18. Springer, 2001.
10. Fatbardh Veseli "HIKOS - Highly Secure, Intelligent Software Copy-Protection." Master's Thesis Master of Science in Information Security Department of Computer Science & Media Technology Gjøvik University College, 2011
11. Jan CAPPAERT "Code Obfuscation Techniques for Software Protection", Dissertation for the degree of Doctor in Engineering, Ghent University, April 2012.
12. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of AES," in *Advances in Cryptology—EUROCRYPT* (LNCS 6632). Berlin, Germany: Springer, 2011, pp. 69–88.
13. Gaurav R Berad, Mayur C Akewar, "Implementation of dedicated Hardware Using Encryption Technology," FIFTH POST GRADUATE CONFERENCE OF COMPUTER ENGINEERING, CPGCON 2016.

BIOGRAPHY

Gaurav Berad I hereby take this opportunity to express my heartfelt gratitude towards the people whose help was very useful to complete my dissertation work.I would like to thank all the faculties who have cleared all the major concepts that were involved in the understanding of the techniques behind my project work.

Prof. Mayur C Akewar I take this opportunity to express my heartfelt gratitude to my guide, Prof. Mayur C Akewar Department of Computer Engineering, RMDSSOE, Savitribai Phule, Pune University, for his kind cooperation and capable guidance during the course of the dissertation, without which the dissertation would have been difficult to proceed with.