ISSN: 2278-4632
Vol-12 Issue-09 No.01 September 2022

DESIGN AND IMPLEMENTATION OF ENCRYPTION/DECRYPTION ARCHITECTURES FOR ADVANCED ENCRYPTION STANDARDS

Shaik Arifa, Student, Department of ECE,QUBA COLLEGE OF ENGINEERING AND TECHNOLOGY,Quba Nagar,Venkatachalam-524320,SPSR Nellore Dist.A.P.,India.
Mr.V.Kiran Kumar, M.Tech, AssistantProfessor, DepartmentofECE, QUBACOLLEGEOFENGINEERINGANDTECHNOLOGY, QubaNagar, Venkatachalam-524320, SPSRNelloreDist. A.P., India.

ABSTRACT

The main objective of this project is to develop an efficient security process. This project presents hardware architecture optimized for accelerating the encryption and decryption operations of the Brakerski/Fan-Vercauteren (BFV) homomorphic encryption scheme with high-performance polynomial multipliers. As an enhancement of this project an efficient secured advanced encryption standard algorithm is introduced for whole improvement.

INTRODUCTION

Cryptography, often called encryption, is the practice of creating and using a cryptosystem or cipher to prevent all but the intended recipient(s) from reading or using the information or application encrypted. A cryptosystem is a technique used to encode a message. The recipient can view the encrypted message only by decoding it with the correct algorithm and keys. Cryptography is used primarily for communicating sensitive material across computer networks. The process of encryption takes a clear-text document and applies a key and a mathematical algorithm to it, converting it into crypto-text. In crypto-text, the document is unreadable unless the reader possesses the key that can undo the encryption. In 1997 the National Institute of Standards and TECHNOLOGY (NIST), a branch of the US government, started a process to identify a replacement for the Data Encryption Standard (DES). It was generally recognized that DES was not secure because of advances in computer processing power. The goal of NIST was to define a replacement for DES that could be used for non-military information security applications by US government agencies. Of course, it was recognized that commercial and other non-government users would benefit from the work of NIST and that the work would be generally adopted as a commercial standard. The NIST invited cryptography and data security specialists from around the world to participate in the discussion and selection process. Five encryption algorithms were adopted for study. Through a process of consensus the encryption algorithm proposed by the Belgium cryptographers Joan Daeman and Vincent Rijmen was selected. Prior to selection Daeman and Rijnmen used the name Rijndael (derived from their names) for the algorithm. After adoption the encryption algorithm was given the name Advanced Encryption Standard (AES) which is in common use today. In 2000 the NIST formally adopted the AES encryption algorithm and published it as a federal standard under the designation FIPS-197. The full FIPS-197 standard is available on the NIST web site (see the Resources section below). As expected, many providers of encryption software and hardware have incorporated AES encryption into their products. The AES encryption algorithm is a block cipher that uses an encryption key and a several rounds of encryption. A block cipher is an encryption algorithm that works on a single block of data at a time. In the case of standard AES encryption the block is 128 bits, or 16 bytes, in length. The term “rounds” refers to the way in which the encryption algorithm mixes the data re-encrypting it ten to fourteen times depending on the length of the key. This is described in the Wikipedia article on AES encryption. The AES algorithm itself is not a computer program or computer source code. It is a mathematical description of a process of obscuring data. A number of people have created source code implementations of AES encryption, including the original authors. AES encryption uses a single key as a part of the encryption process. The key can be 128 bits (16 bytes), 192 bits (24 bytes), or 256 bits (32 bytes) in length. The term 128-bit encryption refers to the use of a 128-bit encryption key. With AES both the encryption and the decryption are performed using the same key. This is called a symmetric encryption algorithm.

Encryption algorithms that use two different keys, a public and a private key, are called asymmetric encryption algorithms. An encryption key is simply a binary string of data used in the encryption process. Because the same encryption key is used to encrypt and decrypt data, it is important to keep the encryption key a secret and to use keys that are hard to guess. Some keys are generated by software used for this specific task.

LITERATURE SURVEY

In the past two decades, Internet of Things (IoT) has emerged as a distinguishable approach and it has attracted the attention of many researchers. Security and Privacy is the major concern in IoT. Lightweight cryptography is the main aspect of security which has led to many results being published in the research literature. The following section presents some research in the field related to IoT. Paulo S.L.M. Barreto and Vincent Rijmen (2000) presented a lightweight SPN block cipher KHAZAD that favors component reuse following WTS [24] (Wide Trail Strategy). In WTS round transformations have two invertible steps; first, local non-linear transformation (any output bit depends upon a limited number of input bits) and second, a linear mixing transformation for achieving high diffusion. KHAZAD have a 64-bits long block and its key size is 128-bits. Its S-box is randomly generated and decryption differs from encryption only in the key schedule [25]. Kazumaro Aoki et al. (2000) presented Camelia, a 128-bit Feistel block cipher having 128/192/256-bit keys with 18/24/24 round for Camelia-128/192/256 respectively. It uses four different 8X8 S-box in the non-linear layer, designed to minimize hardware size with additional input/output key whitening. The F-function uses SPN structure and is inserted after every 6 round [26]. Phillip Rogaway et al. (2001) described a parallelizable block-cipher mode of operation, named OCB for efficient Authenticated Encryption that provides privacy and authenticity. OCB is an advancement over IAPM that performs encryption on arbitrary length bit string $M \in \{0, 1\}*$ using $|M|/n\_ +$ 2block-cipher invocations, where n is the block length. In it offset calculation and session setups are economical and a single cryptographic key is proposed. There is no extended-precision addition and proposed block cipher calls are most favorable [64]. William C. Barker and Elaine Barker (2004) specified TDEA (Triple Data Encryption Algorithm) Feistel block cipher by implementing DEA cryptographic engine. Its block size is 64-bit and key size is also 64-bit (56 bits randomly generated by the algorithm as key bits and remaining 8-bits used for error detection). Operations performed in DEA engine are initial permutation, complex key dependent computation, and inverse initial permutation. DEA engine runs in two directions – forward and reverse. It is the sequence in which the key bits are used that varies in these two directions. In TDEA forward and inverse operation is defined as a compound operation of DEA forward and inverse transformations. TDEA key consists of a key bundle, KEY with three keys. To be secure the number of blocks processed with on bundle key should be less than 232 [27]. Katsuyuki Takashima (2005) designed mCrypton (miniature of Crypton), a lightweight SPN enhancement over Crypton. It has 64-bit block size, 64/96/128-bits key size and number of rounds are 12. Round transformation has four steps: substitution (nibble-wise using four 4-bit S-boxes), bit permutation (Column-wise), transposition (Column-to-Row), and key addition. mCrypton processes 8-byte data block into 4X4 nibble array representation as in Crypton. Key scheduling consists two stages; the first stage is a key generation for round using S-boxes and the second stage is key variable update through round constant and rotations (word-wise and bit-wise). S-boxes used in key scheduling are same as that in round transformation. Decryption differs by encryption with a different key schedule [22]. There exists a MITM attack on mCrypton [28]. Francois-Xavier et al. (2006) designed a Feistel lightweight block cipher, SEA for software implementations on an 8-bit processor. SEAn,b operates on a number of blocks and key sizes. Operations performed in SEA are: bitwise ExclusiveOR; S-box in parallel; word rotation and inverse word rotation; bit rotation and; addition mod. In key scheduling, during half rounds, the master key is encrypted while during the other half rounds master key is decrypted [29]. A. Bogdanov et al. (2007) specified a SPN block cipher PRESENT suitable for hardware implementations. PRESENT has 64-bits block size, 31 rounds and a key size of length 80/ 128 bits. Each round has XOR operation for a round key and a post-whitening key. Each round of PRESENT applies same 4- bit S-

ISSN: 2278-4632
Vol-12 Issue-09 No.01 September 2022

Box 16 times in parallel for non-linear substitution layer that increases confusion. Diffusion by using a bit permutation for the linear diffusion layer [6]. Due to its weak diffusion property different attacks like linear attack, weak key attack and saturation attacks are applied to PRESENT.

CRYPTOGRAPHY

Cryptography is a method of protecting information and communications through the use of codes so that only those for whom the information is intended can read and process it. The pre-fix "crypt" means "hidden" or "vault" and the suffix "graphy" stands for "writing."
In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms to transform messages in ways that are hard to decipher. These deterministic algorithms are used for cryptographic key generation and digital signing and verification to protect data privacy, web browsing on the internet and confidential communications such as credit card transactions and email.

CRYPTOGRAPHIC ALGORITHMS

Cryptosystems use a set of procedures known as cryptographic algorithms, or ciphers, to encrypt and decrypt messages to secure communications among computer systems, devices such as smartphones, and applications. A cipher suite uses one algorithm for encryption, another algorithm for message authentication and another for key exchange. This process, embedded in protocols and written in software that runs on operating systems and networked computer systems, involves public and private key generation for data encryption/decryption, digital signing and verification for message authentication, and key exchange.

HOMOMORPHIC ENCRYPTION

The concept of homomorphic encryption was introduced in [1], of which two of the authors are Ronald L. Rivest and Len Alderman. The R and the A in RSA encryption.The most popular example for the use of homomorphic encryption is where a data owner wants to send data up to the cloud for processing, but does not trust a service provider with their data. Using a homomorphic encryption scheme, the data owner encrypts their data and sends it to the server. The server performs the relevant computations on the data without ever decrypting it and sends the encrypted results to the data owner. The data owner is the only one able to decrypt the results, since they alone have the secret key.

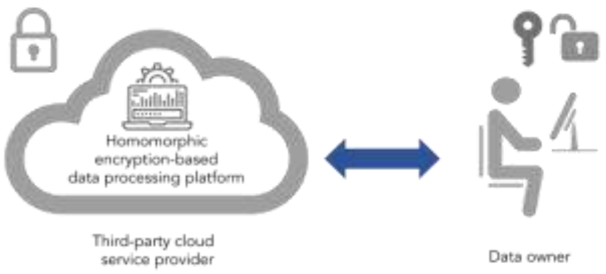


Figure:Homomorphic encryption scheme

So far, the most efficient homomorphic encryption scheme when performing the same operations on multiple ciphertexts at once is the Brakerski-Gentry-Vaikuntanathan (BGV) [11] scheme. The Brakerski/Fan-Vercauteren (BFV) [2, 3] and the Cheon-Kim-Kim-Song (CKKS) [4] schemes share second place for efficiency. BGV is, however, more difficult to use. These are all lattice-based cryptographic schemes dependent on the hardness of the Ring Learning With Errors (RLWE) problem, which (at least for now) means that they are quantum-safe.
BGV, BFV, and CKKS are additively and multiplicatively homomorphic (i.e., you can perform both addition and multiplication within the encrypted domain). No division. No exponentiating a number by an encrypted one. No non-polynomial operations. In BGV and BFV, computations can only be performed on integers. In CKKS, computations can be performed on complex numbers with limited precision.

Homomorphic Encryption Libraries

There are five open source homomorphic encryption libraries that I've heard good things about:

1. PALISADE
2. SEAL
3. HElib
4. HEAAN
5. TFHE

Of these five, I've personally used PALISADE, SEAL, and HElib. All three implement the BFV scheme. SEAL and HElib both implement CKKS as well.

PALISADE is a DARPA and IARPA-funded project (previously NSA-funded too). It was developed and is supported by a superbly talented team at the New Jersey Institute of Technology (Gerard Ryan, Professor YuriyPolyakov, and Professor Kurt Rohloff).

SEAL was developed by the Microsoft Research Cryptography Research Group.

HElib was developed by ShaiHalevi and Victor Shoup, both esteemed figures in the cryptographic community. The library does, however, have less support than PALISADE and SEAL do.

TFHE implements a scheme that is faster than BGV for individual operations, but which cannot make use of the SIMD method mentioned above.

PALISADE, HElib, and SEAL can be freely used within commercial applications, with each using an NJIT license, Apache v2.0 license, and MIT license respectively.

EXISTING TECHNIQUE

Fan and Vercauteren extendBrakerski's encryption scheme from learning with errors (LWE) setting to ring LWE (RLWE) setting. The RLWE problem is simply a ring-based version of the LWE problem , which leads to the following encryption scheme as described in . Let the plaintext and ciphertext spaces be the rings Rt and Rq , respectively, for some integers q > t > 1. We remark that neither q nor t have to be primes nor that t and q have to be coprime. flooring, round to nearest integer, and the reduction by modulo q operations, respectively. Let also a $ ←− S indicate that a is uniformly sampled from the set S, χ be a truncated discrete Gaussian distribution, and =q/t. For the rest of this article, n, q, and t denote the degree of polynomial modulus, the prime used as ciphertext modulus, and the integer used as plaintext modulus, respectively. Secret and public key generation and encryption and decryption operations described in textbook-BFV are shown in the following.

Algorithm 1 Modified Iterative NTT Algorithm

```
Input: $a(x) \in \mathbb{Z}_q[x]/(x^n + 1)$
Input: primitive $n$-th root of unity $\omega \in \mathbb{Z}_q$, $n = 2^l$
Output: $\overline{a}(x) = \mathrm{NTT}(a) \in \mathbb{Z}_q[x]/(x^n + 1)$
for $i$ from 1 by 1 to $l$ do
    $m = 2^{l-i}$
    for $j$ from 0 by 1 to $2^{i-1} - 1$ do
        for $k$ from 0 by 1 to $m - 1$ do
            $curr\_\omega = \omega[2^{i-1}k]$
            $U \leftarrow a[2 \cdot j \cdot m + k]$
            $V \leftarrow a[2 \cdot j \cdot m + k + m]$
            $a[2 \cdot j \cdot m + k] \leftarrow U + V$
            $a[2 \cdot j \cdot m + k + m] \leftarrow \omega \cdot (U - V)$
        end for
        $\omega \leftarrow \omega \cdot \omega_i$
    end for
end for
return $a$
```

PROPOSED METHOD

ADVANCED ENCRYPTION STANDARDS

The Advanced Encryption Standard, in the following referenced as AES, is the winner of the contest, held in 1997 by the US Government, after the "Data Encryption Standard" was found too weak

because of its small key size and the technological advancements in processor power. Fifteen candidates were accepted in 1998 and based on public comments the pool was reduced to five finalists in 1999. In October 2000, one of these five algorithms was selected as the forthcoming standard: a slightly modified version of the Rijndael.
The Rijndael, whose name is based on the names of its two Belgian inventors, Joan Daemenand Vincent Rijmen, is a Block cipher, which means that it works onfixed-length group of bits, which are called blocks. It takes an input block of a certain size,usually 128, and produces a corresponding output block of the same size. Thetransformation requires a second input, which is the secret key. It is important to knowthat the secret key can be of any size (depending on the cipher used) and that AES usesthree different key sizes: 128, 192 and 256 bits.
To encrypt messages longer than the block size, a mode of operation is chosen, which we will explain at the very end of this tutorial, after the implementation of AES. While AES supports only block sizes of 128 bits and key sizes of 128, 192 and 256 bits, the original Rijndael supports key and block sizes in any multiple of 32, with a minimum of 128 and a maximum of 256 bits.

DESCRIPITION OF AES ALGORITHM

AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The different transformations operate on the intermediate results, called state. The state is a rectangular array of bytes and since the block size is 128 bits, which is 16 bytes, the rectangular array is of dimensions 4x4. (In the Rijndael version with variable block size, the row size is fixed to four and the number of columns varies. The number of columns is the block size divided by 32 and denoted Nb). The cipher key is similarly pictured as a rectangular array with four rows. The number of columns of the cipher key, denoted Nk, is equal to the key length divided by 32.
A state:

```
-----------------------------
| a0,0 | a0,1 | a0,2 | a0,3 |
| a1,0 | a1,1 | a1,2 | a1,3 |
| a2,0 | a2,1 | a2,2 | a2,3 |
| a3,0 | a3,1 | a3,2 | a3,3 |
-----------------------------
```

A key:

```
-----------------------------
| k0,0 | k0,1 | k0,2 | k0,3 |
| k1,0 | k1,1 | k1,2 | k1,3 |
| k2,0 | k2,1 | k2,2 | k2,3 |
| k3,0 | k3,1 | k3,2 | k3,3 |
-----------------------------
```

It is very important to know that the cipher input bytes are mapped onto the state bytes in the order a0,0, a1,0, a2,0, a3,0, a0,1, a1,1, a2,1, a3,1 ... and the bytes of the cipher key are mapped onto the array in the order k0,0, k1,0, k2,0, k3,0, k0,1, k1,1, k2,1, k3,1 ... At the end of the cipher operation, the cipher output is extracted from the state by taking the state bytes in the same order. AES uses a variable number of rounds, which are fixed: A key of size 128 has 10 rounds. A key of size 192 has 12 rounds. A key of size 256 has 14 rounds.
During each round, the following operations are applied on the state:

1. Sub Bytes: every byte in the state is replaced by another one, using the Rijndael S-Box
2. Shift Row: every row in the 4x4 array is shifted a certain amount to the left
3. Mix Column: a linear transformation on the columns of the state.
4. AddRoundKey: each byte of the state is combined with a round key, which is a different key for each round and derived from the Rijndael key schedule.

In the final round, the Mix Column operation is omitted. The algorithm looks like the following (pseudo-C):

ISSN: 2278-4632
Vol-12 Issue-09 No.01 September 2022

```
AES (state, Cipher Key)
{
Key Expansion (Cipher Key, Expanded Key);
AddRoundKey (state, Expanded Key);
For (i = 1; i< Nr; i++)
{
Round (state, Expanded Key + Nb*i);
}
Final Round (state, Expanded Key + Nb * Nr);
}
```

Features of AES

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

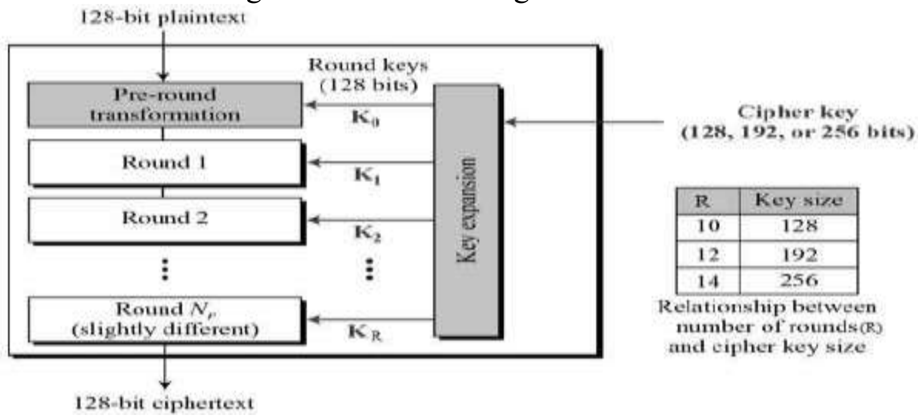
Operation of AES

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration −



R	Key size
10	128
12	192
14	256

Figure:AES process

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −

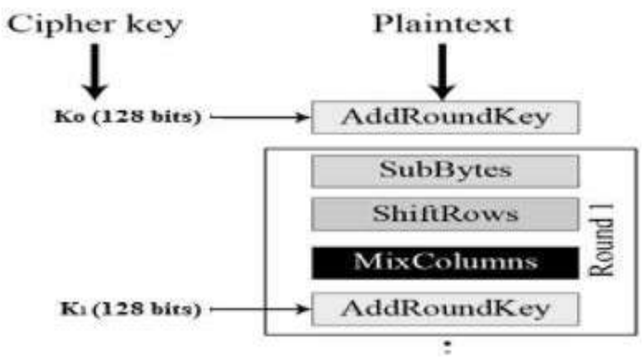


Figure: key expansion

Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

AES Analysis

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

In this chapter, we will discuss the different modes of operation of a block cipher. These are procedural rules for a generic block cipher. Interestingly, the different modes result in different properties being achieved which add to the security of the underlying block cipher.

A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

XILINX

Xilinx Tools is a suite of software tools used for the design of digital circuits implemented using Xilinx Field Programmable Gate Array (FPGA) or Complex Programmable Logic Device (CPLD). The design procedure consists of (a) design entry, (b) synthesis and implementation of the design, (c) functional simulation and (d) testing and verification. Digital designs can be entered in various ways using the above CAD tools: using a schematic entry tool, using a hardware description language (HDL) – VHDL or VHDL or a combination of both. In this lab we will only use the design flow that involves the use of VHDL HDL.

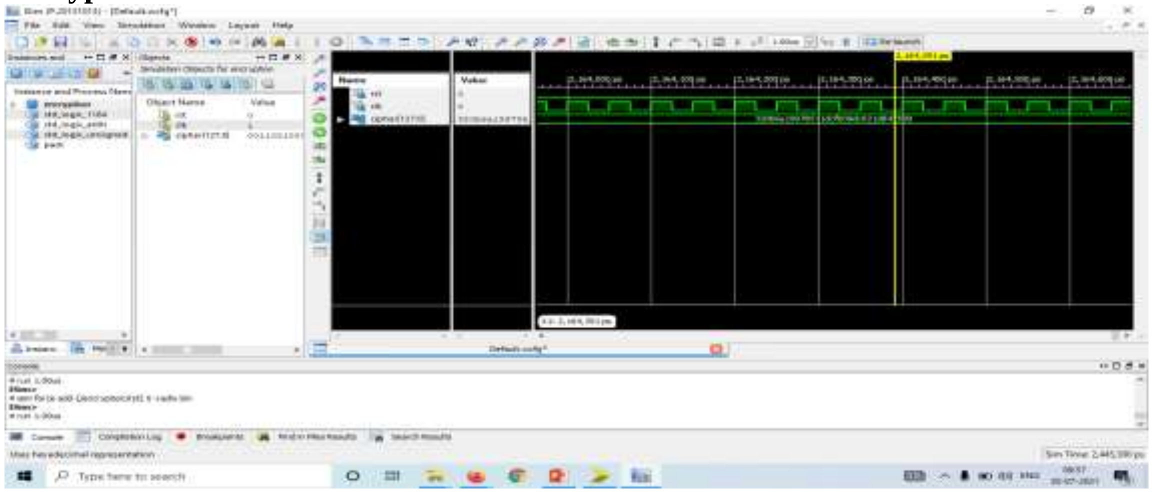
The CAD tools enable you to design combinational and sequential circuits starting with VHDL HDL design specifications. The steps of this design procedure are listed below:

1. Create VHDL design input file(s) using template driven editor.
2. Compile and implement the VHDL design file(s).
3. Create the test-vectors and simulate the design (functional simulation) without using a PLD (FPGA or CPLD).
4. Assign input/output pins to implement the design on a target device.
5. Download bitstream to an FPGA or CPLD device.
6. Test design on FPGA/CPLD device

A VHDL input file in the Xilinx software environment consists of the following segments:
Header: module name, list of input and output ports.
Declarations: input and output ports, registers and wires.
Logic Descriptions: equations, state machines and logic functions.
End: endmodule All your designs for this lab must be specified in the above VHDL input format. Note that the state diagram segment does not exist for combinational logic designs.

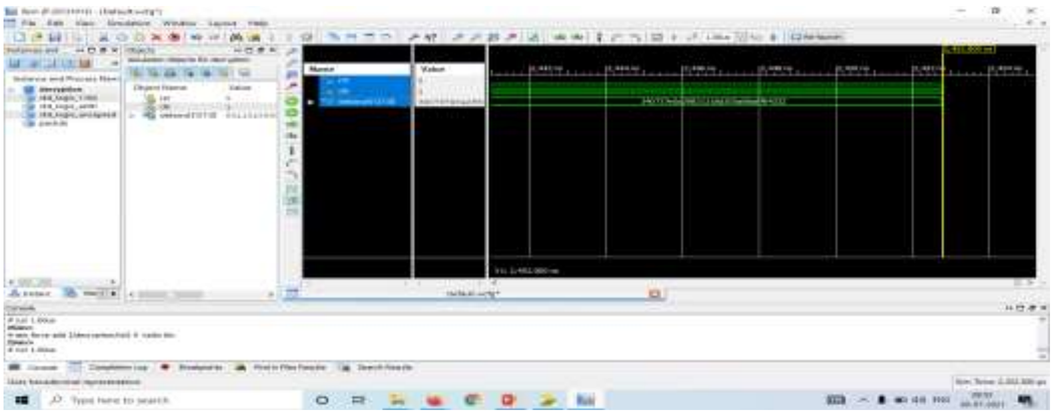
RESULT

Encryption



ISSN: 2278-4632
Vol-12 Issue-09 No.01 September 2022

Decryption



CONCLUSION

Crypto may be seen as a continuous struggle between cryptographers & cryptanalysts. Attacks on cryptography have an equally long history. The security of cryptographic modules for providing a practical degree of protection against white-box (total access) attacks should be examined in a totally un-trusted execution environment.

So many developers design so many devices to protect the data very powerful when it is done right, but it is not a panacea. But by using this crypto devices technique we are providing secure scan architecture can easily be integrated into the scan-based DFT design flow as the synthesis register can be specified to the corresponding bit of the secret key. The secure control circuit & multiplexers between the MKR & secret key can be inserted.

In this project a solution is presented that consists in using an AES-based cryptographic core commonly embedded in secure system. Three addition modes are added to the current mission of the AES crypto core. One for pseudo- random test pattern generation & one for signature analysis. Efficiency of these three modes has been demonstrated. Extra cost in terms of area is very low compared to other techniques. Because only one AES core will be originally embedded in the system. This reduces the reduction of test cost will lead to the reduction of overall production cost & 100% security of data.

FUTURE ENHANCEMENT

In this project we are designing a crypto devices with low complexity and high security which having the data and key length of 128.futher we can extend the keyand data upon to 192 and 256 bits efficiently and successfullyby using the same technique.

REFERENCES

1. S. Reddy, “Easily testable realizations for logic functions,” IEEE Transactions on Computers, vol. 21, no. 11, pp. 1183–1188, 1972.
2. S. Golomb, Shift Register Sequences. Aegean Park Press, 1982.
3. R. K. Brayton, C. McMullen, G. Hatchel, and A. Sangiovanni-Vincentelli, Logic Minimization Algorithms For VLSI Synthesis. Kluwer Academic Publishers, 1984.
4. E. McCluskey, “Built-in self-test techniques,” IEEE Design and Test of Computers, v Vol. 2, pp. 21–28, 1985.
5. D. H. Green, “Families of Reed-Muller canonical forms,” International Journal of Electronics, vol. 70, pp. 259–280, 1991.
6. M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design. Jon Willey and Sons, New Jersey, 1994
7. H.-J. Wunderlich, “BIST for systems-on-a-chip,” Integration, the VLSI Journal, vol. 26, no. 1-2, pp. 55 – 78, 1998.

8. M.G. Kuhn, R.J. Anderson. Soft tempest: hidden data transmission using electromagnetic emanations. Information Hiding 1998,LNCS 1525,pp.124-142,1998.
9. D. Bleichenbacher. Chosen Cipher text Attacks against Protocols Based on the RSA Encryption Standard PKCS #1. CRYPTO'98, LNCS 1462, pp.1-12, 1998.
10. K.Gandolfi,C.Mourte,F. Olivier. Electromagnetic Analysis: Concrete Results. CHES 2001,LNCS 2162,pp.251-261, 2001.