International Journal of Electrical Electronics & Computer Science Engineering
Volume 6, Issue 6 (December, 2019) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

FPGA Design Implementation of AES and RSA for Information Security

Swati Malik
Assistant Professor, Department of ECE, Maharaja Surajmal Institute of Technology, New Delhi, India
swatimalik@msit.in

Abstract: ***With recent advancements in the field of semiconductor industry, information security turns out to be major concern especially for the applications related to cryptography. Rivest- Shamir Adleman(RSA) and Advanced Encryption Standard (AES) are the two popularly identified encryption schemes that guarantees authenticity and confidentiality of the data over an insecure communication channel. This paper compares the performance of 64-bit RSA and AES algorithm using Verilog HDL. Experimental results shows that AES turns out to be stronger option in terms of processing speed, whereas in terms of security RSA proves to be highly Efficient and avoids usage of third party for key exchange. Result analysis reveals that AES stands out to be better option as it is affordable in terms of security and possess higher processing speed and low power consumption.***

Keywords: Advanced Encrytion Standard (AES), Rivest-Shamir Adleman (RSA), Cryptography, Verilog HDL, Cipher Key.

I. INTRODUCTION

The vital requirement for data security is hiding information from extraneous public or malicious invaders. This requirement has given rise to various cryptographic techniques has given birth to different kinds of cryptographic techniques comprising of asymmetric and symmetric cryptography, message authentication codes, digital signatures, hash function etc. In symmetric encryption technique, sharing of key takes place between sender and receiver which is kept undisclosed from the invader. Among the different variants of symmetric algorithms, Advanced Encryption Standard (AES) is getting popular due to its better efficiency and better security than its antecedents [1]. It was first acknowledged by Federal Information Processing Standard (FIPS) 192, which got published in November 2001 [2, 3]. AES uses a secret key to encrypt and decrypt any information and functions on 128-bit fixed block. AES may possibly be organized to use 3 diverse key lengths and thereafter the algorithms are titled AES-128, AES-192 and AES-256 respectively to represent the bit length of the key. Contrasting to the symmetric cryptography, asymmetric cryptography make use of a pair of keys in order to encrypt and decrypt any information. Out of these two keys one key is known as public key (distributed to others) whereas, other one is called private key which is meant to be kept secret. Normally, the information is encrypted using a public key which can be decrypted through private key only. Few essential properties of asymmetric cryptography [4] that are to satisfied are:

- Computationally efficient key generation process should be used.
- Public key of the receiver for any information is used to compute the cipher text for the sender.
- Cipher should be decrypted at the receiver's end text by using his own private key.
- It is difficult or at least unrealistic to determine the private key from the analogous public key.

It is computationally inefficient to compute the plain text form the public key and cipher text.RSA is the most commonly used asymmetric encryption technique which was invented by Ronald Rivest, Adi Shamir, and Len Adleman in the year 1977 [5]. As it is a public key encryption technique, the private key is kept secret but the public key is publicized to everyone in RSA. Since its invention, RSA is considered as one of the most secure cryptosystems existing in communication systems. The purpose of this paper is to describe the basic encryption and decryption method as well as to cover the hardware level implementation using Verilog HDL for the two most widely used encryption schemes.

II. OVERVIEW OF AES CRYPTOSYSTEM

AES i.e. Advanced Encryption Standard 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 [6].

Advanced Encryption Standard is a cryptographic technique that works on matrices. The linear data is first converted into a matrix, eg. 128 bit data is converted into a 4x4 matrix, each block of 1 byte each. We can have

International Journal of Electrical Electronics & Computer Science Engineering
Volume 6, Issue 6 (December, 2019) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

three different types of input plaintext in AES i.e 128 bit, 192 bits and 256 bit. For each type of input we use different numbers of processing rounds. The table below shows the relation between the size of plaintext and the number of rounds of processing. Initially AES algorithm operations are performed on a two dimensional array of bytes called states [7]. The basic unit for processing in the AES algorithm is a byte. Steps to be followed are given below:

1. Add cypher key: initiate the algorithm by adding the cipher key matrix to the plaintext matrix i.e XOR operation is performed between each block of the two matrices.

2. Loop runs till number of rounds minus 1: Loop iterates till N-1 where N is the number of rounds of processing.

2.(a) Substitute Bytes: The first step to a round is to do a byte by byte substitution with a lookup table called an S-box. An S-box is a one to one mapping for all byte values from 0 to 255. The S-box is used to change the original plain text in bytes to cipher text.

2.(b) Shift Rows: The next step in the round is to shift the rows of the state. The rows are shifted x number of bytes to the left where x is the row number. This means row 0 will not be shifted, row 1 will be shifted 1 byte to the left, row 2 will be shifted 2 bytes to the left, and row 3 will be shifted 3 bytes to the left. Note that the shifts do not affect the byte substitution so the shifting and S-box operations could be done in a different order.

2.(c) Mix Column: After applying the S-box and shifts to the state the operation of a Mix Column is used. The Mix Column lookup table takes a byte and transforms it into four bytes. The Mix Column table is generated by the following algorithm. Each element in the table consists of four bytes usually represented as hexadecimal values. The second and third bytes are always same as the input byte.

2.(d) Add round Key: After mixing column round key is added to the final matrix. Then after this addition the process goes on for N-1 times where N is the number of rounds.

3. Final Round

3.(a) Substitute Bytes

3.(b)Shift Rows

3.(c)Add round key

III. Overview Of RSA Cryptosystem

RSA is named after the initial letters of surnames of Ron Rivest, Adi Shamir and Leonard Adleman who first publicly announced the algorithm in 1978. RSA is public key cryptosystem that is used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, and if the public key is large enough, only someone with knowledge of the prime numbers can decode the message feasibly. Breaking RSA encryption is known as the RSA problem [8, 9]. Whether it is as difficult as the factoring problem remains an open question. Prime numbers are used as product of two 32 bit prime numbers is that it is difficult to prime factorize such a large number. Even the computer decoder can take years to do the prime factorization when the number increases to such a length. The graph show the difference in prime factorization of two numbers, one small and the other large enough to use be used in RSA.

Most important part of RSA algorithm is key generation. The steps are as follows:

- Select two random prime numbers p and q
- Calculate $n = p \times q$
- Calculate $\emptyset(n) = (p-1) \times (q-1)$
- Select integer e such that $\gcd(\emptyset(n), e) = 1; 1<e<\emptyset(n)$; where $e \& \emptyset(n)$ are relatively prime.
- Calculate $d = (e-1)\,(mod\ \emptyset(n))$

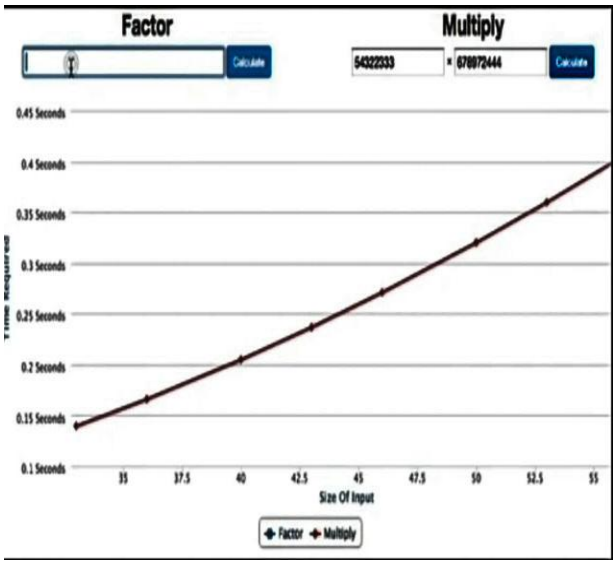


Fig. 1. Time Taken Vs Size of Prime Number (Small)

International Journal of Electrical Electronics & Computer Science Engineering
Volume 6, Issue 6 (December, 2019) | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

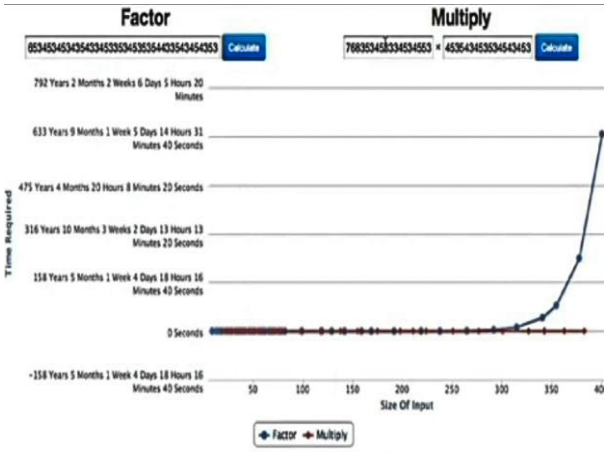


Fig. 2. Time Taken Vs Size of Prime Number (Large)

According to the procedure the encryption key e is available but the decryption key d is not known to all. Mathematically this procedure is defined as, M is the actual message, C is the converted message or cipher text by using publicly available encryption key e, and d is the decryption key [10].

$$C = M \times e \ (mod\ m)$$

$$M = C \times d \ (mod\ m)$$

RSA encryption and decryption are mutual inverses and commutative. Generation of two numbers uses 32 bit Pseudo Random number generator i.e. LFSR. LFSR stands for Linear Feedback Shift Register which is generally used to generate pseudo random numbers in digital circuits. An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops. Linear feedback shift registers make extremely good pseudorandom pattern generators. When the outputs of the flip-flops are loaded with a seed value (anything except all 0s, which would cause the LFSR to produce all 0 patterns) and when the LFSR is clocked, it will generate a pseudorandom pattern of 1s and 0s. Note that the only signal necessary to generate the test patterns is the clock [11].

When using a primitive polynomial, maximum length of an LFSR sequence is 2n-1.A 32-bit LFSR will produce a sequence of over 4 billion random bits, or 500 million random bytes. The polynomial used for generating this sequence of 32-bit pseudorandom numbers is as under:

$$P(x) = x^{32} + x^{22} + x^2 + x + 1$$

- After generating the two 32 bit numbers from the LFSR we first neglect the even number generated as we are looking for a prime number [12].
- When we get the two odd numbers we check for their primality. This is done using the Miller Rabin Primality Test. The normal way is not used as it makes the process slow.

IV. RESULTS AND DISCUSSION

The AES and RSA algorithms are evaluated using Verilog HDL. It is simulated using Xilinx ISE Design Suite. For RSA peak memory usage was 361 Mb whereas for AES it is390 Mb. Maximum power dissipation for RSA cryptography technique is 45mW but for AES it is 41mW.

Table I. Hardware Utilization of RSA Cryptosystem

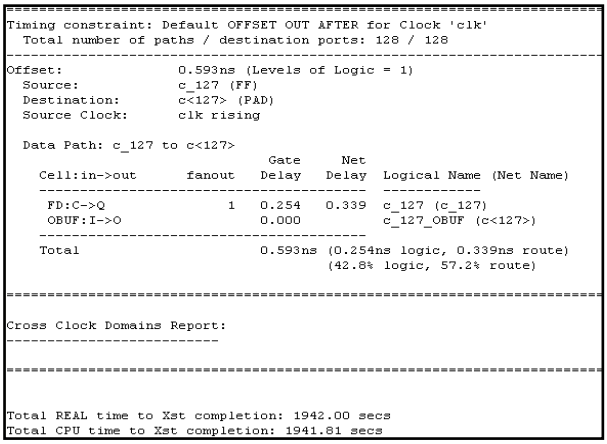
Category	Used	Available	Usage %
Slice LUT's	981	41,000	2%
Occupied Slices	339	10250	3%
Bonded IOB	73	300	24%

```
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 24 / 24
--------------------------------------------------------------------------------
Offset:              0.593ns (Levels of Logic = 1)
  Source:            c_23 (FF)
  Destination:       c<23> (PAD)
  Source Clock:      clk rising

  Data Path: c_23 to c<23>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q               1   0.254   0.339  c_23 (c_23)
     OBUF:I->O                 0.000          c_23_OBUF (c<23>)
    ----------------------------------------
    Total                      0.593ns (0.254ns logic, 0.339ns route)
                                       (42.8% logic, 57.2% route)

==========================================================================

Cross Clock Domains Report:
--------------------------


==========================================================================


Total REAL time to Xst completion: 74.00 secs
Total CPU time to Xst completion: 73.30 secs
```

Fig. 3. Net Delay and Offset for RSA

Table II. Hardware Utilization of AES Cryptosystem

Category	Used	Available	Usage %
Slice LUT's	6292	41,000	15%
Occupied Slices	1788	10250	17%
Bonded IOB	257	300	85%



```
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 128 / 128
--------------------------------------------------------------------------------
Offset:              0.593ns (Levels of Logic = 1)
  Source:            c_127 (FF)
  Destination:       c<127> (PAD)
  Source Clock:      clk rising

  Data Path: c_127 to c<127>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q               1   0.254   0.339  c_127 (c_127)
     OBUF:I->O                 0.000          c_127_OBUF (c<127>)
    ----------------------------------------
    Total                      0.593ns (0.254ns logic, 0.339ns route)
                                       (42.8% logic, 57.2% route)

==========================================================================

Cross Clock Domains Report:
--------------------------

==========================================================================


Total REAL time to Xst completion: 1942.00 secs
Total CPU time to Xst completion: 1941.81 secs
```

Fig. 4. Net Delay and Offset for AES

V. CONCLUSION

Implementation of RSA circuit is done using Verilog HDL. The corresponding table for both the algorithms is mentioned and the two compared on several basis such as no. of slice LUTs used and bonded IOB, and the outputs are mentioned for comparison of processing speed.After going through the results, we find that the in terms of processing speed AES appears to be the stronger option. However, in terms of security RSA proves highly efficient, and avoids the use of third party for key exchange.In all, to choose the best among the good AES stands out to be a better option as it is affordable in terms of security and has a much higher processing speed and power consumption issues.

VI. REFERENCES

[1] Joan Daemen, Steve Borg and Vincent Rijmen. The Design of Rijndael: AES The Advanced Encryption Standard. Springer, 2002.

[2] A. J. Elbirt,W. Yip, B. Chetwynd, C. Paar. An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 9:545–557, 2001.

[3] T. Kumaki, M. Yoshikawa, and T. Fujino, “Cipher-destroying and secret-key-emitting hardware trojan against aes core,” in 2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS), Aug 2013, pp. 408–411.

[4] D. Hely, F. Bancel, M.-L. Flottes, and B. Rouzeyre, “Secure scan techniques: a comparison,” in On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International. IEEE, 2006, pp. 6–pp.

[5] Adi Shamir Ronald Rivest and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21:120–126, 1978.

[6] S. S. Ali, O. Sinanoglu, S. M. Saeed, and R. Karri, “New scan-based attack using only the test mode,” in Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on. IEEE, 2013, pp. 234–239.

[7] B. Roy, R. P. Giri, A. C., and B. Menezes, “Design and implementation of an espionage network for cache-based side channel attacks on aes,” in 2015 12th International Joint Conference on e-Business and Telecommunications (ICETE), vol. 04, July 2015, pp. 441–447.

[8] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury, “An efficient approach to develop secure scan tree for crypto- hardware,” in Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on. IEEE, 2007, pp. 21–26.

[9] N. Ryuta, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, “Scan-based side-channel attack against rsa cryptosystems using scan signatures,” IEICE transactions on fundamentals of electronics, communications and computer sciences, vol. 93, no. 12, pp. 2481– 2489, 2010.

[10] D. Mukhopadhyay, S. Banerjee, D. Roy Chowdhury, and B. B. Bhattacharya, “Cryptoscan: A secured scan chain architecture,” in Test Symposium, 2005. Proceedings. 14th Asian. IEEE, 2005, pp. 348–353.

[11] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, “Securing scan design using lock and key technique,” in 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT’05), Oct 2005, pp. 51–62.

[12] Flevina Jonese D’souza, Dakshata Panchal, "Advanced Encryption Standard (AES) Security Enhancement using Hybrid Approach", IEEE International Conference on Computing Communication and Automation (ICCCA), 2017.