Design Modulo Multiplier for Symmetric Key Cryptography Using HDL

(IJCSNT) International Journal of Communication Systems and Network Technologies
ISSN Number: 2053-6283
Volume 11, Issue No. 1, 10–18

https://journals.mirlabs.in/index.php/ijcsnt
DOI-10.18486/ijcsnt/11.1.143
IJCSNT

Sachin Singh[1], Laxmi Singh[2], Poonam Sinha[3], Sanjeev Gupta[4]

Abstract

This paper covers the design of modulo multiplier for International Data Encryption Algorithm (IDEA) cryptography. The current era has seen an explosive growth in communications. Applications like online banking, personal digital assistants, mobile communication, smartcards, etc. have emphasized the need for security in resource constrained environments. International Data Encryption Algorithm (IDEA) cryptography serves as a perfect network security tool because of its 128 bits key sizes and high security comparable to that of other algorithms. However, to match the ever increasing requirement for speed in today's applications, hardware acceleration of the cryptographic algorithms is a necessity.

Keywords

IDEA, Crptography, Mobile Communication, Modulo Multiplier

Received: 01 December 2021; **Revised:** 31 March 2022; **Accepted:** 27 April 2022;
Published: 30 April 2022;

[1]Department of Electronics & Communication Engineering, Rabindranath Tagore University, Raisen, Madhya Pradesh, India.
[2]Department of Electronics & Communication Engineering, Barkatullah University, Bhopal, Madhya Pradesh, India.

Corresponding author:
Email: sachin1388@gmail.com

Introduction

In the field of networking, role of network intrusion detection is immense. It is a very vital tool which provides the security against various external and internal threats in any network. To understand the theory of network intrusion detection, the understanding and knowledge of different threats in the network. To maintain network intrusion detection in a network involves the fulfilment of the security goals in the network which are Data Confidentiality, Integrity, Authentication and Non-Repudiation. Data confidentiality is achieved by means of Cryptography.

International Data Encryption Algorithm (IDEA) is a block cipher designed by Xuejia Lai and James L. Massey of ETH-Zürich and was first described in 1991. It is a minor revision of an earlier cipher, PES (Proposed Encryption Standard); IDEA was originally called IPES (Improved PES). IDEA was used as the symmetric cipher in early versions of the Pretty Good Privacy cryptosystem.

IDEA was to develop a strong encryption algorithm, which would replace the DES procedure developed in the U.S.A. in the seventies. It is also interesting in that it entirely avoids the use of any lookup tables or S-boxes. When the famous PGP email and file encryption product was designed by Phil Zimmermann, the developers were looking for maximum security. IDEA was their first choice for data encryption based on its proven design and its great reputation.

The IDEA encryption algorithm

- provides high level security not based on keeping the algorithm a secret, but rather upon ignorance of the secret key
- is fully specified and easily understood
- is available to everybody
- is suitable for use in a wide range of applications
- can be economically implemented in electronic components (VLSI Chip)
- can be used efficiently
- may be exported world wide
- is patent protected to prevent fraud and piracy.

In this paper, the cipher used is a symmetric key block cipher. It takes its input as 64 bit plain text and gives a 64 bit cipher text as output using a 128 bit key. While working on plain text, it divides the input data in to 16 bit sub-blocks and operates on each block. It is described as one of the more secure block algorithm due to its high immunity to attacks. In spite of the fact that Data Encryption standard (DES) is another popular symmetric block cipher which is used in several financial and business application and its drawback is the short key word length. Moreover unlike DES, IDEA doesn't need any S-box or P-box is required for implementing this cipher. The most crucial module part of this algorithm is the design of the multiplier modulo a Fermat prime, which is one of the algebraic group operation used and entire speed of IDEA depends on this module. So designing the multiplier is a major during the hardware or software implementation of IDEA because its speed is a big issue when hardware implemented IDEA is used in real time application. The overall objective for hardware implementation of IDEA is to minimize the hardware requirements which result in efficient use of silicon area and at the same time improve the processing speed and high throughput of data. As the performance of IDEA cipher depends entirely on the modulo(2n+1) multiplier design, the main objective is to design an efficient and fast modulo multiplier which is to be used in the entire IDEA algorithm.

The organization of the rest of the paper is as follows. The previous hardware and software implementations are covered in section II. Section III describes the IDEA cipher and its detailed operations as well as modules. Section IV describes the general architecture of the cryptosystem to be implemented and the proposed modulo multiplier architecture. Section V discusses the performance reviews and comparisons with previous schemes and section VI finally concludes the paper.

Previous Work

In spite of the fact that IDEA works with 16 bit word blocks, software implemented IDEA cannot reach the speed that is required for online encryption in high speed networks. IDEA was implemented in software by Ascom, the patent holder of IDEA, and it achieved an encryption rate of 23.53 Mbps.[1]

The IDEA Algorithm

The proposed Encryption Standard (PES) is a block cipher introduced by Lai and Massey.[2] It was then improved by the Lai, Massey and Murphy in 1991. This version, with stronger security against differential analysis and truncated differentials, was called the Improved PES (IPES). IPES was renamed to be the International Data Encryption Algorithm (IDEA) in 1992. Claims have been made that the algorithm is the most secure block encryption algorithm in the public domain. IDEA is a symmetric, secret-key block cipher. The keys for both encryption and decryption must be kept secret from unauthorized persons. Since the two keys are symmetric, one can divide the decryption key from the encryption one or vice versa. The size of the key is fixed to be 128 bits and the size of the data block which can be handled in one encryption/decryption process is fixed to 64 bits. All data operations in the IDEA cipher are in 16-bit unsigned integers. When processing data which is not an integer multiple of 64-bit block, padding is required. The security of IDEA algorithm is based on the mixing of three different kinds of algebraic operations: EX-OR, addition and modular multiplication. IDEA is based upon a basic function, which is iterated eight times. The first iteration operates on the input 64-bit plain text block and the successive iterations operate on the 64-bit block from the previous iteration. After the last iteration, a final transform step produces the 64-bit cipher block.

The decryption phase of IDEA is identical to that of the encryption phase.

It uses the same sequence of operations as in the encryption phase. The only change is that the sub-keys are reversed and are slightly different. That means the sub-keys which are used in round 1 during encryption phase are manipulated during last round of decryption phase. The sub keys used in decryption are either additive or multiplicative inverse of the sub-keys used in the encryption phase.

Key Generation

The 64-bit plaintext block is partitioned into four 16-bit sub-blocks, since all the algebraic operations used in the encryption process operate on 16-bit numbers. Another process produces for each of the encryption rounds, six 16-bit key sub-blocks from the 128-bit key. Since a further four 16-bit key-sub-blocks are required for the subsequent output transformation, a total of 52(= 8 x 6 + 4) different 16-bit sub-blocks have to be generated from the 128-bit key.

The 52 16-bit key sub-blocks which are generated from the 128-bit key are produced as follows:

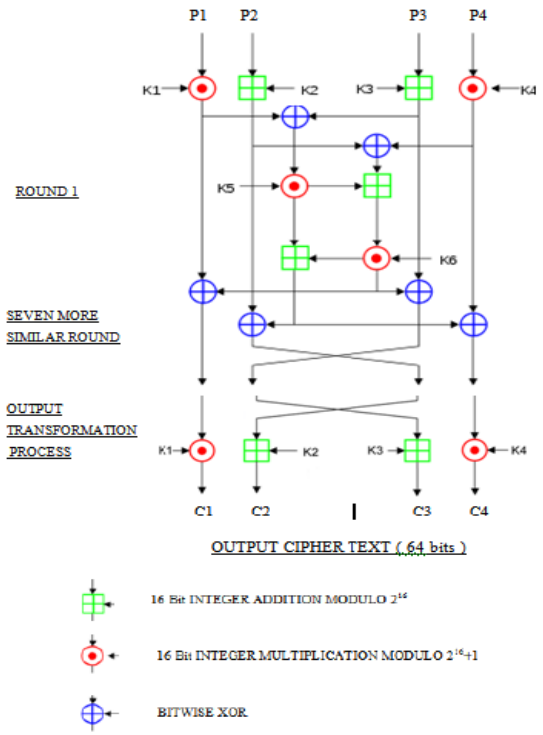

Figure 1. Basic structure of IDEA Algorithm

- First, the 128-bit key is partitioned into eight 16-bit sub-blocks which are then directly used as the first eight key sub-blocks.
- The 128-bit key is then cyclically shifted to the left by 25 positions, after which the resulting 128-bit block is again partitioned into eight 16-bit sub-blocks to be directly used as the next eight key sub-blocks.
- The cyclic shift procedure described above is repeated until all of the required 52 16-bit key sub-blocks have been generated.

Design And Implementation

The main objective of implementing a cryptosystem is to increase the data encryption and decryption rate so as to increase the throughput. FPGA implementations of cryptosystem offer adequate speed so that it can be easily embedded in real time applications.[3] To implement an algorithm in hardware, we have to first realize the architecture of the entire system. The general architecture of hardware implementation of a cryptosystem is shown in Figure 3.

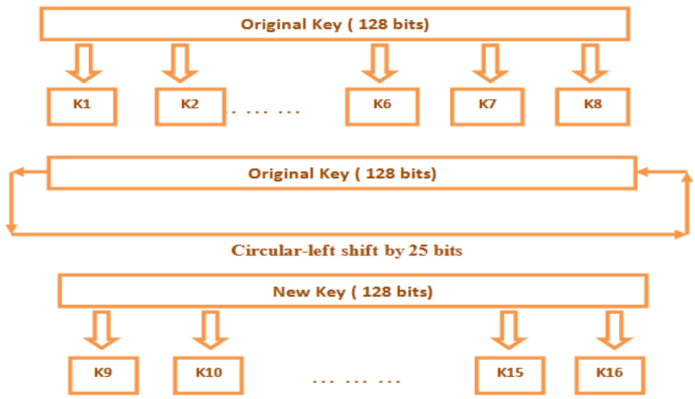

Figure 2. IDEA Algorithm Key generation

Hardware Implementation of IDEA

The IDEA cipher has three separate units other than key generation module. The performance and speed of hardware implemented IDEA depends on these three modules. These modules are:

- Multiplier Module.
- Addition unit.
- Inverse modulo Multiplier.

In each round of the 8 rounds of algorithm, the following sequences of events are performed:

1. Multiply* P1 and K1
2. Add* P2 and K2
3. Add* P3 and K3
4. Multiply* P4 and K4
5. XOR the results of step 1 and step 3
6. XOR the results of step 2 and step 4
7. Multiply* the results of step 5 with K5
8. Add* the results of step 6 and step 7
9. Multiply* the results of step 8 with K6
10. Add* the results of step 7 and step 9
11. XOR the results of step 1 and step 9
12. XOR the results of step 3 and step 9
13. XOR the results of step 2 and step 10
14. XOR the results of step 4 and step 10

Sequence of events followed in the output transformation round:-

1. Multiply* R1 and K1
2. Add* R2 and K2
3. Add* R3 and K3
4. Multiply* R4 and K4

Among these modules, the main component which controls the speed and performance of IDEA is the modulo $(2^n + 1)$ multiplier module.[4] It consumes the major portion of the clock cycles required by the entire algorithm. The modulus used in the multiplication is a Fermat prime which is $(2^n + 1)$. One important thing here is that the operand 0 is treated as 2^n. The implementation of this multiplier in hardware is the most difficult task because the word length of the operands is comparatively large and implementing the multiplication sequentially is really time consuming.

Multiplication Modulo $(2^n + 1)$

Multiplication modulo a Fermat prime (p) is used as an important operation in many algorithms and it is crucial in various applications like pseudorandom number generation, Arithmetic processing and Cryptography.[5] In IDEA algorithm, this modulo multiplier plays a very important role in the throughput and speed. In general, a modulo multiplier consists of two stages, Multiplier module and modulo reduction. This paper mainly deals with the various multiplication schemes that have been implemented along with some newly proposed schemes.

The multiplier module is the stage where two binary n bit numbers are multiplied to form a product of 2n bits[5]. The modulo reduction stage produce the product modulo the Fermat prime number and the final output becomes an n bit number[6–8]. Various implementations have been done on this multiplier module so as to improve the efficiency of the cryptosystem[9]. The most crucial part of multiplying two binary numbers is the generation of partial products[10]. A lot of problems arise when two numbers are multiplied in a straightforward approach in hardware. As per human nature of calculation, when any expression is given as

$$xy \bmod (2^n + 1) = z' \bmod (2^n + 1) = z \tag{1}$$

At first the product is calculated by traditional method and then the modulo reduction is done by iterative subtraction method until the value falls under the range of 0 and 2_ .But the drawbacks of this implementation is inefficient use of silicon area which increases the hardware cost and it is time consuming[11].

Proposed Modulo Multiplier

There are several multipliers existing for idea[12,13]. Here we are presenting a new kind of modulo multiplier which is highly optimized as compared to previous one. The description for proposed multiplier are as follows. In general if we multiply the two n bit number ,then we get 2n bit number. Store this 2n bit number(result) in to temporary register ie.t1,then make the length of modulo(2ˆn+1) equal to the length of 2n bit number ie.t1, by consented zeroes(‘0’s) after the LSB bit of the modulo(2ˆn+1) and store this value into t2. Here we defining the subsequent steps by RTL diagram, which is easy to understand[14].

We show the difference between normal modulo multiplier and the proposed multiplier[15]. In normal method:

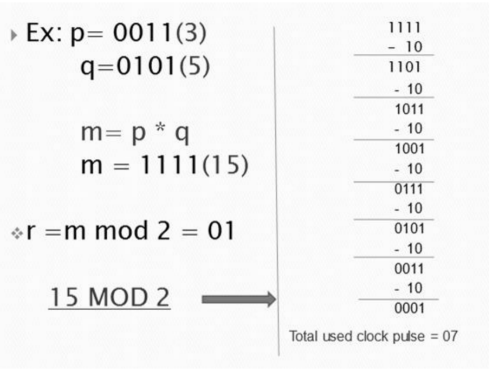

Figure 3. Normal modulo multiplier method

Here p=3, and q= 5 , when multiply these two number then answer is m=15,the final output of the modulo multiplier is r=m mod 2^n=1,in this example compare the time taken by normal & proposed modulo multiplier[16–19].

In proposed method:

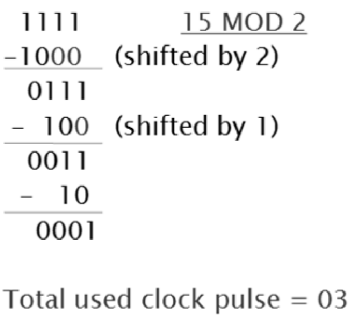

Figure 4. Proposed modulo multiplier method

When compare the both method then the time taken by proposed multiplier is very less[20,21].

Results And Observations

The performance parameters which are to be accounted for implementing IDEA in hardware are:

- **Throughput or data conversion rate**: It is taken as an important tool for measuring the timing performance of IDEA i.e. the amount of data processed and encrypted per unit of time[22].

- **Area Requirements**: It can be reported either in terms of number of Look-Up Tables (LUTs) or number of slices[23].
- **Maximum Clock frequency**: It is the maximum clock frequency that can be achieved by the design.
- **Latency**: It denotes the delay i.e. the time taken for the input data to move to the output port. For a pipelined design, the latency is measured by product of delay of a single pipelined stage and the number of pipelined stages[24].

Conclusion

RSA Security goes on to say that IDEA was analyzed to measure its strength against differential cryptanalysis. The analysis concluded that IDEA is immune to that technique. In fact, there are no linear cryptanalytic attacks on IDEA, and there are no known algebraic weaknesses in IDEA. The only weakness of note was discovered by Daemen: using any of a class of 2^{51} weak keys during encryption results in easy detection and recovery of the key. However, since there are 2^{128} possible keys, this result has no impact on the practical security of the cipher for encryption provided the encryption keys are chosen at random. IDEA is generally considered to be a very secure cipher and both the cipher development and its theoretical basis have been openly and widely discussed.

References

[1] Zimmermann R, Curiger A, Bonnenberg H et al. A 177mb/s vlsi implementation of the international data encryption algorithm. *IEEE Journal of Solid-State Circuits* 1994; 29: 303–307.

[2] Lai X and Massey JL. A proposal for a new block encryption standard. In *Advances in Cryptology – EUROCRYPT 90*. Berlia, Germany: Springer Verlag, pp. 389–404.

[3] Ranjan R and Poonguzhali I. Vlsi implementation of idea encryption algorithm. In *Mobile and Pervasive Computing (CoMPC–2008)*.

[4] Modugu R, Kim YB and Choi M. A fast low-power modulo 2n + 1 multiplier. *Journal of IET Computers & Digital Techniques* 2011; .

[5] Timarchi S and Navi K. Improved modulo 2n + 1 adder design. *International Journal of Computer and Information Engineering* 2008; 2(7).

[6] Hämäläinen A, Tommiska M and Skyttä J. 6.78 gigabits per second implementation of the idea cryptographic algorithm. Springer-Verlag, pp. 760–769.

[7] Leong MP, Cheung OYH, Tsoi KH et al. A bit serial implementation of the international data encryption algorithm idea. *IEEE* 2000; .

[8] Kitsos P, Sklavos N, Galanis MD et al. 64 bit blockciphers: Hardware implementations and comparison analysis. *Elsevier* 2004; : 593–604.

[9] Modugu R, Kim YB and Choi M. Design and performance measurement of efficient idea crypto-hardware using novel modular arithmetic components. In *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*. pp. 1222–1227.

[10] Thaduri M, Yoo S and Gaede R. An efficient implementation of idea encryption algorithm using vhdl. *Elsevier* 2004; .

[11] Michalski A, Gaj K and El-Ghazawi T. An implementation comparison of an idea encryption cryptosystem on two general-purpose reconfigurable computers ; .

[12] Dharmapurikar S and Lockwood J. Fast and scalable pattern matching for network intrusion detection systems. *IEEE Journal on Selected Areas in Communications* 2006; 24: 1781–1792.

[13] Chiranth E, Chakravarthy HVA, Naga mohana reddy P et al. Implementation of rsa cryptosystem using verilog. *International Journal of Scientific & Engineering Research* 2011; 2(5).

[14] Shirali M, Tefke T, Staudemeyer RC et al. A survey on anonymous communication systems with a focus on dining cryptographers networks. *IEEE Access* 2023; 4: 1–30.

[15] Meraouche I, Dutta S, Tan H et al. Neural networks-based cryptography: A survey. *IEEE Access* 2021; 9: 124727–124740.

[16] Pelosi G, Selleri S and Florence. A leap in cryptography: The leon battista alberti cypher disk. *IEEE* 2021; : 7–11.

[17] Gabriel A, Camargo M, Monticolo D et al. Improving the idea evaluation process in creative workshops through contextualization. *Journal of Cleaner Production* 2016; 135: 1503–1513.

[18] Huang QX, Yap WL, Chiu MY et al. Privacy-preserving deep learning with learnable image encryption on medical images. *IEEE Access* 2022; 10: 66345–66355.

[19] Almasri O and Jani HM. Introducing an encryption algorithm based on idea. *International Journal of Science and Research (IJSR)* 2013; 2(9): 334–339.

[20] Javed Y, Khan AS, Qahar A et al. Preventing dos attacks in iot using aes. *Journal of Telecommunication, Electronic and Computer Engineering* ; 9(3-11): 55–60.

[21] Revathi Bai P and Sarala B. System on chip implementation of pipelined based advanced encryption standard and rand shifter for secure memory. *Journal of Engineering Physics* 2021; 12(10): 134–141.

[22] Hämäläinen P, Alho T, Hännikäinen M et al. Design and implementation of low-area and low-power aes encryption hardware core. *Tampere University of Technology / Institute of Digital and Computer Systems* ; : 1–7.

[23] Zodpe H and Sapkal A. An efficient aes implementation using fpga with enhanced security features. *Journal of King Saud University – Engineering Sciences* 2020; 32: 115–122.

[24] Smekal D, Hajny J and Martinasek Z. Comparative analysis of different implementations of encryption algorithms on fpga network cards. *IFAC PapersOnLine* 2018; 51-6: 312–317.