Volume 2, No. 7, July 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA) – A TYPICAL ILLUSTRATION

Sandipan Basu
Department of Computer Science, Asutosh College, Calcutta University,
Kolkata-700026, West Bengal, INDIA
mail.sandipan@gmail.com

Abstract: There are several symmetric and asymmetric data encryption algorithms. IDEA (International Data Encryption Algorithm) is one of the strongest secret-key block ciphers. In this article, I try to represent the existing IDEA algorithm in a different way. In the following illustration, we would see how the encryption can be expressed in a simpler way.

Keywords*:* Round, Output Transformation, Sub-Key, Symmetric Key Algorithm.

INTRODUCTION

International Data Encryption algorithm (IDEA) is a block cipher algorithm designed by Xuejia Lai and James L. Massey of ETH-Zürich and was first described in 1991.The original algorithm went through few modifications and finally named as International Data Encryption Algorithm (IDEA). The mentioned algorithm works on 64-bit plain text and cipher text block (at one time). For encryption, the 64-bit plain text is divided into four 16 bits sub-blocks. In our discussion, we denote these four blocks as P1 (16 bits), P2 (16 bits), P3 (16 bits) and P4 (16 bits). Each of these blocks goes through 8 ROUNDS and one OUTPUT TRANSFORMATION phase. In each of these eight rounds, some (arithmetic and logical) operations are performed. Throughout the eight ROUNDS, the same sequences of operations are repeated. In the last phase, i.e., the OUTPUT TRANSFORMATION phase, we perform only arithmetic operations. At the beginning of the encryption process, the 64 bit plain text is divided in four equal size blocks and ready for ROUND1 input. The output of ROUND1 is the input of ROUND2. Similarly, the output of ROUND2 is the input of ROUND3, and so on. Finally, the output of ROUND8 is the input for OUTPUT TRANSFORMATION, whose output is the resultant 64 bit cipher text (assumed as C1 (16bits), C2 (16 bits), C3 (16 bits) and C4 (16 bits)). As the IDEA is a symmetric key algorithm, it uses the same key for encryption and for decryption. The decryption process is the same as the encryption process except that the sub keys are derived using a different algorithm [6]. The size of the cipher key is 128bits. In the entire encryption process we use total 52 keys (ROUND1 to ROUND8 and OUTPUT TRANSFORMATION phase); generated from a 128 bit cipher key. In each round (ROUND1 to ROUND8) we use six sub keys. Each sub-key consists of 16bits. And the OUTPUT TRANSFORMATION uses 4 sub-keys.

DETAILED ANALYSIS IDEA

As we mentioned before, in the IDEA algorithm, we take input text of size 64bits at a time and divide it in evenly; i.e., 64bit plain text is divided into 4 sub-blocks, each of 16bits in size.

Now, let us look, what are the basic operations needed in the entire process.

Operations needed in the first 8 rounds -

1. Multiplication modulo $2^{16}+1$.
2. Addition modulo 2^{16}.
3. Bitwise XOR.

And, operations needed in the OUTPUT TRANSFORMATION phase –

1. Multiplication module $2^{16}+1$.
2. Addition modulo 2^{16}.

All the above mentioned operations are performed on 16 bit sub-blocks. For simplicity of expressing the operations, we denote, Multiplication modulo $2^{16}+1$ by * symbol, and Addition modulo 2^{16} by, + symbol. And bitwise XOR will be represented by its usual symbol $\oplus$.

Now, let us take a look on the key generation for the encryption process. Using 25-bit circular left shift operation on the original key, we produce other subsequent sub-keys, used in different rounds. For instance, among the total no. of 52 keys- Sub-key K1 is having first 16bits of the original key, sub-key K2 is having the next 16 bits, and so on till sub-key K6; i.e., for ROUND1, sub-keys K1 to K6 use first (16x6=) 96 bits of the original cipher key.

In ROUND2, sub-key K7 & K8 take the rest of the bits (bits 97 to 128) of the original cipher key. Then we perform circular left shift (by 25bits) operation on the original key. As a result the 26^{th} bit of the original key shifted to the first position and becomes the first bit (of the new shifted key) and the 25^{th} bit of the original key, moves to the last position and becomes the 128^{th} bit (after first shift).

This process continues till ROUND8, and also in the OUTPUTPUT TRANSFORMATION phase; i.e., after the ROUND8, the key is again shifted left by 25 bits and the first 64 bits of the shifted key is taken for use, and used as sub-keys K49 to K52 in the OUTPUT TRANSFORMATIONMATION phase. So, form the above observations, we could write –

Table 1. Sub-Keys & corresponding Bit positions

Bit Positions	Sub-keys
1-96	ROUND1 (K1 to K6) ; ROUND5 (K25 to K30) and *OT* (K49 to K52) [Bit position 1-64]
97-128 & 1-64 (of shifted key)	ROUND2 (K7 to K12) ; ROUND6 (K31 to K36)
65-128 & 1-32 (of shifted key)	ROUND3 (K13 to K18) ; ROUND7 (K37 to K42)
33-128	ROUND4 (K19 to K24) ; ROUND8 (K43 to K48)

[OT: OUTPUT TRANSFORMATION]

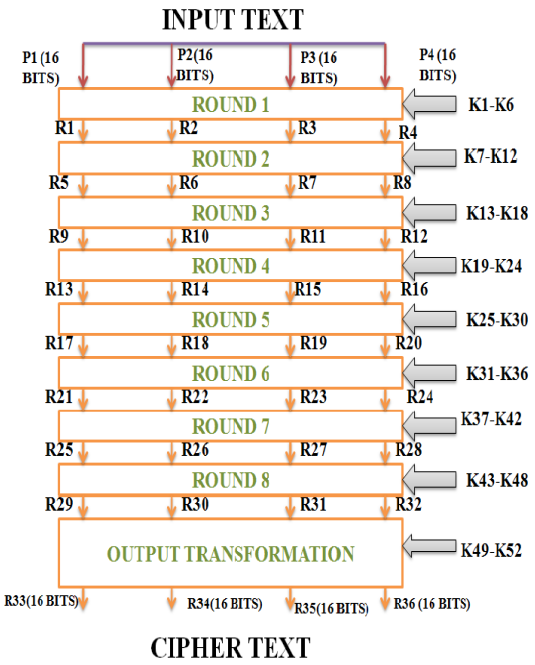

Figure 1. Overview of IDEA algorithm

ANALYSIS OF IDEA

Let us assume, in our example, in Fig.1, the four initial blocks are P1, P2 P3 and P4. That is, P1, P2, P3 and P4 are the input to ROUND1. Assume, the outputs of ROUND1 are, R1, R2, R3 and R4. Similarly, the output of ROUND2, denoted as R5, R6, R7 and R8; and so on. Previously, we have seen, that what are the operations taking place in one round of IDEA. Now, we analyze the outputs of ROUND1. If we have a close look on ROUND1 (or any these 8 rounds), then we would observe, that, all the operations that are taking place in a round, can eventually be expressed as simple equations. The outputs of ROUND1 can be written as below –

R1= P1 * K1⊕ [[{(P2 + K2)⊕ (P4 * K4)} + [{(P1 * K1)⊕ (P3 + K3)} * K5]] * K6]

R2= P3 + K3⊕ [[{(P2 + K2)⊕ (P4 * K4)} + [{(P1 * K1)⊕ (P3 + K3)} * K5]] * K6]

R3= P2 + K2 ⊕[{(P1 * K1) ⊕(P3 + K3)} *K5] + *[[{(P2 + K2)⊕(P4 * K4)} + [{(P1 * K1)⊕(P3 + K3)} * K5]] * K6]]*

R4= P4 * K4 ⊕[{(P1 * K1) ⊕(P3 + K3)}* K5] + *[[{(P2 + K2)⊕(P4 * K4)} + [{(P1 * K1)⊕(P3 + K3)} * K5]] * K6]]*

The outputs of other subsequent rounds can also be written in the same manner. From the above we would notice, to calculate R1, R2, R3 and R4, we have to perform a lot of operations. But, in each round of IDEA, we are performing the same operations again & again and also unnecessarily. For instance, to calculate R1, we perform bitwise XOR operation with the result of (P1 * K1) and the underlined portion (as shown in the equation). Again, to calculate R2 we perform the same operations again as underlined in R1. To calculate R3 and R4, we perform the same (indicated as italic+underlined) operations again and again. Apart from this, in the calculation of R3 and R4, again we perform the same sequence of operations (the entire underlined portions of R3 and R4).

Hence, there is no logic to perform the same set of operations again & again for R1, R2, R3 and R4.

For simplicity we just perform each of the basic operations for once, and use its result for other further calculations. The basic required operations that are involved in calculating R1, R2, R3 and R4, are as follows -

1. P1 * K1
2. P4 * K4
3. P2 + K2
4. P3 + K3

Other operations are based on these basic operations.

Next, we represent a block diagram of a hardware unit, needed to implement IDEA algorithm. To implement it in hardware, we need some separate hardware components to accomplish the individual tasks, and as a whole too.

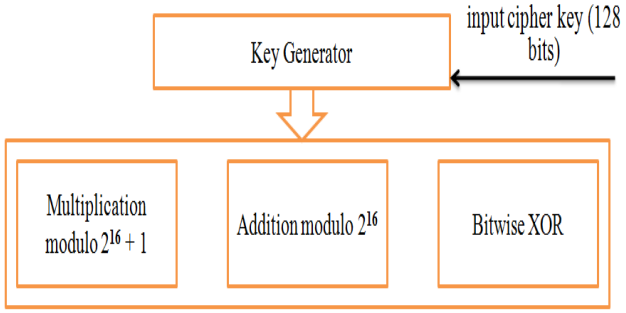

Figure 2: Hardware components required for IDEA

In Fig.2, we would see, that, there is a unit named "Key generator". At the beginning of the encryption process, we provide the original (128bits) cipher key to the mentioned unit. When necessary, the KEY GENERATOR unit produces different sub-keys by performing circular left shift operation (by 25bits) on the current key and provides the sub-keys to other units (multiplication modulo 2^{16} +1 and addition modulo 2^{16} units). The unit named as "Multiplication modulo 2^{16} + 1", is used to perform all the multiplication modulo 2^{16}+1 operation, when required. The same is for unit "Addition modulo 2^{16}" and unit "Bitwise XOR".

Now, we look forward for the parallel implementation of IDEA algorithm. In our approach, we have shown, the entire encryption process can be performed in several steps and performing operations in parallel wherever possible. Parallelism in operations can be achieved both in software and using hardware. In the computation of R1, R2, R3 and

R4 the above mentioned operations can be performed in the following way, at different time instants -

Time Unit 1: *t1*=P1 * K1, *t3*=P3 + K3

(These two operations can be done in parallel by their corresponding units.t1 & t3 are used to hold the temporary results.)

Time Unit 2: *t2*=P2 + K2; *t4*=P4 * K4; *t5*=t1 ⊕*t3*

Time Unit 3: *t6*=t2 ⊕t4; *t7*=t5 * K5

Time Unit 4: *t8*=t6 + t7

Time Unit 5: *t9*=t8 * K6

Time Unit 6: *R1* = t9 ⊕t1; *t10*=t9+t7

Time Unit 7: *R2* = t9 ⊕t3

Time Unit 8: *R3* = t10 ⊕t2

Time Unit 9: *R4* = t10 ⊕t4;

From the above illustration, it is noticeable, if we want to perform the operations of one round in parallel, then it would take at least 9 time units. Operations written in single time unit can be done in parallel. We could also see, after the completion of ROUND1, the partial encrypted input text R1 is produced at time unit 6, R2 at time unit 7, R3 at time unit 8 and R4 at time unit 9.

But, for the next round, i.e., for ROUND2, the computation R5 would involve the usage R4 (generated from **ROUND1**). So, before we start the computation for ROUND2, we will have to wait 9 time units; i.e., until R4 is generated from ROUND1. Hence, till ROUND 7, all the partial encrypted cipher texts (R1 to R28) are generated, and would take at least (7 rounds x 9) = 63 time units. Now, let us look how we can improve ROUND 8 & the OUTPUT TRASNFORMATION phase. For the following illustration, we would assume, sub-keys K49-K52 are available in the ROUND 8 also. So, the computation for ROUND 8 and OUTPUT TRANSFORMATION can also be accomplished as follows -

Time Unit 64: *t1*=R25 * K43, *t3*=R27 + K45

Time Unit 65: *t2*=R26 + K44; *t4*=R28 * K46; *t5*=t1 ⊕*t3*

Time Unit 66: *t6*=t2 ⊕t4; *t7*=t5 * K47

Time Unit 67: *t8*=t6 + t7

Time Unit 68: *t9*=t8 * K48

Time Unit 69: *R29* = t9 ⊕t1; *t10*=t9+t7

Time Unit 70: *R30* = t9 ⊕t3; *R33* = R29 * K49

Time Unit 71: *R31* = t10 ⊕t2; *R34* = R30 + K50

Time Unit 72: *R32* = t10 ⊕t4; *R35* = R31 + K51

Time Unit 73: *R36* = R32 * K52

Form the above illustration, we can see, if sub-keys K49 to K51 are available in the ROUND8, then we will be able to perform, the first three operations of OUTPUT TRANSFORMATION in ROUND8. Hence, the entire process can be made faster.

So, to complete the IDEA encryption process, the time required is 73 time units.

DISCUSSIONS

In the above discussion, it is seen, the minimum time taken to complete the encryption process of IDEA is 73 time units. Here, what is the span of each time unit, it depends on the implementation. In software implementation, it would depend on, how efficient code is being written to accomplish parallelism in the operations. In hardware implementation, how fast the encryption process is done, depends on the different circuitry available, the hardware architecture used to achieve parallelism (since, there are many options are available) and also the technology used to design and implement the entire hardware unit (Fig. 2). In implementation on the XCV300-6, the bit parallel version achieved an encryption rate of 1166Mb/sec using an 82MHz clock, whereas the bit serial implementation achieved a 600Mb/sec throughput at a clock rate of 150MHz [2]. The bit-parallel implementation achieved a higher throughput with lower latency than the bit serial implementation, while the bit-serial implementation permits a minimal area fully-parallel design [2]. In the implementation on XCV1000E-6, the total time taken is 1.246μs with maximum clock rate 105.9MHz, with throughput 6.78Gbps [3]. In his paper, Daemen mentioned large classes of weak keys for IDEA. These keys are weak in the sense that it takes only a very small amount of effort to detect their use. It is possible to eliminate the weak key problem by slightly modifying the key schedule of IDEA [4].

CONCLUSION

The IDEA (International Data Encryption Algorithm) is a strong block-cipher. Though there are many operations involved in the entire algorithm, only three different of operations are involved (as mentioned above). As the cipher key size is 128bits, in that respect IDEA is too strong (having taken care for weak keys).

REFERENCES

[1] Chang H.S., "International Data Encryption Algorithm" CS-627-1 Fall, 2004.

[2] Cheung O.Y.H., Tsoi K.H., Leong P.H.W., and Leong M.P. "Tradeoffs in Parallel and Serial Implementations of the International Data Encryption Algorithm IDEA".

[3] Daemen J., Govaerts R., Vandewalle J., "Weak Keys for IDEA", Springer-Verlag, 1998.

[4] Hämäläinen A., Tommiska M., and Skyttä J., "Gigabit per Second Implementation of the IDEA Cryptographic Algorithm", Springer-Verlag Berlin Heidelberg, 2002.

[5] Kahate A., "CRYPTOGRAPHY AND NETWORK SECURITY", Tata-McGraw-Hill, 2nd edition, 2008.

[6] Schneider B., "Applied Cryptography", John Wiley & Sons, Second ed., 1996.

I am Sandipan Basu, M.Sc. (CS).Currently, I am a Guest Faculty, Department of Computer Science, Asutosh College, Kolkata, West Bengal, INDIA.

