DOI: 10.21917/ijct.2010.0022

FPGA BASED HARDWARE KEY FOR TEMPORAL ENCRYPTION

B. Lakshmi[1], E. Kirubakaran[2] and T.N. Prabakar[3]
[1]*Jayaram College of Engineering and Technology, Tiruchirappalli, India*
Email: nmc_lakshmi@yahoo.com
[2]*Bharat Heavy Electricals Ltd, Tiruchirappalli, India*
Email: ekiru@bheltry.co.in
[3]*Saranathan College of Engineering, Tiruchirappalli, India*
Email: tnprabakar@gmail.com

Abstract
In this paper, a novel encryption scheme with time based key technique on an FPGA is presented. Time based key technique ensures right key to be entered at right time and hence, vulnerability of encryption through brute force attack is eliminated. Presently available encryption systems, suffer from Brute force attack and in such a case, the time taken for breaking a code depends on the system used for cryptanalysis. The proposed scheme provides an effective method in which the time is taken as the second dimension of the key so that the same system can defend against brute force attack more vigorously. In the proposed scheme, the key is rotated continuously and four bits are drawn from the key with their concatenated value representing the delay the system has to wait. This forms the time based key concept. Also the key based function selection from a pool of functions enhances the confusion and diffusion to defend against linear and differential attacks while the time factor inclusion makes the brute force attack nearly impossible. In the proposed scheme, the key scheduler is implemented on FPGA that generates the right key at right time intervals which is then connected to a NIOS – II processor (a virtual microcontroller which is brought out from Altera FPGA) that communicates with the keys to the personal computer through JTAG (Joint Test Action Group) communication and the computer is used to perform encryption (or decryption). In this case the FPGA serves as hardware key (dongle) for data encryption (or decryption).

Keywords:
Encryption, Decryption, Real Time Systems, Time Based Key, Brute Force attack, Cryptanalysis, FPGA

1. INTRODUCTION

Cryptography is an art of information security that protects information from unauthorized or accidental disclosure while the information is in transit (either electronically or physically) or in storage, using two processes called encryption (data locking with the key) and decryption (data unlocking with the key). The basic and common cryptanalysis technique an intruder tries is brute force attack. In this case, the attacker tries every possible key on a piece of cipher text until an intelligible translation into plaintext is obtained. It involves traversing the search space of all possible keys until the correct key is found. For symmetric key ciphers, a brute force attack typically means a search of the entire key space in order to recover the plain text from the cipher text. In brute force attack, the expected number of trials before the correct key is found is equal to half of the size of the key space. Information security demands increase in the strength of existing encryption algorithms due to rapid improvements in processing capabilities. Meanwhile the encryption algorithms are effectively implemented both on software and hardware platforms. Even though the software implementation of encryption algorithms has the advantages of portability, flexibility, and ease of use, it provides a limited physical security and agility compared to hardware implementations. The advantages of hardware implementation include less power consumption, small circuit size, hardware reconfiguration, cost efficiency, high operating speed and security. Implementation of software algorithms as hardware is achieved using Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs).The reconfigurable FPGA is economically feasible to perform cryptanalysis compared to ASICs due to their non recurring engineering costs. The use of embedded FPGAs for some dedicated applications has decreased the level of security of encryption algorithms. These devices serve as special purpose hardware and are used to break ciphers. This is because these devices have a hardware implementation of the software. The high level of fabrication techniques and much importantly the concurrency in processing data is used to exploit the security holes through brute force attacks.

The conventional encryption algorithms suffer from the following drawbacks:

1. In conventional encryption algorithms, the algorithmic strength purely depends on the key size used for encryption (or decryption).
2. The conventional algorithms are susceptible to brute force attack and the cryptanalysis is purely dependent on the speed of the system used for cryptanalysis.
3. The linear cryptanalysis is feasible in most of the cases because almost all algorithms possess a linear relationship between inputs and outputs except S-box functions in DES.
4. Every cryptographic algorithm includes a set of weak keys that further weakens the algorithm.
5. The same sequence of operations is to be repeated for a predetermined number of rounds irrespective of the significance of data.
6. The continuous increase in the key size results in making the cryptanalysis difficult and also leads to computational and resources overhead.
7. The invention of dedicated embedded devices like FPGA or ASIC further reduces the time taken for cryptanalysis as they provide a hardware implementation of the software.

To overcome the above issues, a scheme is designed in which the cryptanalysis using brute force attack is made independent of the speed of the system used and the time taken to break the algorithm is increased. To achieve this, time is taken as a second dimension of the key and both data validation and time validation are performed for successful decryption. This involves real time processing and the scheme is also implemented in FPGA that emulates real time processing. Also the key based function selection results in enhanced degree of confusion and diffusion. In the proposed algorithm, the encryption (or decryption) is done on a computer. However, the key scheduler needed for a real time operation is implemented on FPGA and hence, accuracy to the level of nano seconds is achieved.

2. PREVIOUS WORKS

Symmetric Cryptosystems are based on algorithms in which identical keys are used for both encryption and decryption [`1]. Substitution and permutation network (SPN) structure is one of the most widely used structures in block ciphers. The SPN structure is based on Shannon's principles of confusion and diffusion [2] and these principles are implemented through the use of substitution and linear transformation respectively. This confusion and diffusion determines the strength of an algorithm. Many researchers have tried out various enhancements of conventional encryption algorithms. An enhancement approach which combines the advantages of Elliptic Curve Cryptography (ECC) and DES is suggested [3] since in DES the key can be easily revealed and in ECC the process efficiency is low. The algorithm can be used in Computer Supported Cooperative Work (CSCW).

An algorithm [4] based on the difficulty in factoring composite integer into its component primes is named as matrix based asymmetric bulk encryption algorithm. It is estimated that the algorithm is faster than conventional RSA algorithm.

An encryption algorithm based on the application of Optimal Alphabetic Trees (OATs) is introduced [5]. The algorithm presented a theorem to define a family of weight lists that all have the same OAT of a given weight list. The theorem allows finding the levels of cipher text as easier compared to that of plain text. Many authors have suggested methods to increase the strength of DES by changing its internal structure. It is suggested that [6] DES can be modified to use key-dependent S-boxes. Their suggestions improve the cipher's strength against differential, linear, and improved Davies' attacks as well as exhaustive key search.

The key size of 128 bits can offer highest security with today's system but in near future as the speed of processing is continuously increasing, the key size has to be increased to protect the data. Increasing the key size from 80 bits to 128 bits dramatically increases the amount of effort to guess the key. It is said that if the key size is 40 bits and if a system (typical desktop computer) can search 1000 keys per second, then a brute attack requires 35 years to crack the key. Also A computer that searches a billion keys per second, and a billion of these computers, would still take 10,783 billion years to search all possible 128-bit keys [7].

The first exhaustive DES key search machine estimation was proposed and contained 10^6 DES chips, with an estimated cost of 20Million dollars and a 12-hour expected search time [8]. An initial attack was proposed to IDEA [10] based on differential cryptanalysis against up to 2.5 rounds running faster than an exhaustive search. Then a differential- linear attack was presented against 3 rounds and a truncated differential attack on 3.5 rounds. It is possible to break 4.5 rounds of IDEA using impossible differentials [11]. Use of FPGA devices result in enhanced performance measures such as less power consumption, allocation of resources, re-configurability, architecture efficiency and cost efficiency [12]. COPACOBANA is the only reconfigurable parallel FPGA machine optimized for code breaking tasks reported in the open literature which was used to break DES with 120 FPGAs. Depending on the actual algorithm, the parallel hardware architecture can outperform conventional computers by several orders of magnitude [13]. Therefore an algorithm is designed by incorporating time as a second dimension of the key and using dynamically varying sequence of operations. So that the complexity of cryptanalysis increases and the brute force attack and is independent of the speed of the system used for cryptanalysis. The proposed time based scheme is a novel base algorithm that uses a two dimensional element time to increase the complexity of cryptanalysis [9].

3. PREVIOUS IMPLEMENTATION

3.1. ENCRYPTION ALGORITHM OF BASE SYSTEM

The base algorithm was developed in 'C'.

3.1.1 Procedure:

1. 128 bit data& 128-bit key (64 data bits & 64 bits for time) is read.
2. Generation of random number 'r' for the number of rounds, each data set is to be operated.
3. The first data set is operated 'r' times in 'n' available functions.
4. The order of function selection is again based on a random number generated. Example: Assume there are 4 functions available. A random number between 1 & 25 calls first function, 26 & 50 calls second function and so on.

Function 1: Rotate the data by 32 bits and the resultant is 'XOR'ed with 64 bit key.

Function 2: Odd number bits of data bits are 'XOR'ed with even number bits of key and vice versa.

Function 3: A standard S box substitution is performed.

The order of the function calls is maintained in a log, which is also encrypted in the same way.

Hence, the log file will be consisting of

1. 'r' – number of rounds each data set is operated.
2. ABACBAB – order of function calls, Function 1 is denoted by A and so on. (in this case r='7'). (This makes the number of rounds each data set operated to vary dynamically)
3. The result after the 'r' rounds is written as cipher text.

Since, this log is also encrypted using fixed number of rounds and fixed functions and is embedded in the cipher text. A part of the key is used to locate the position of the log file in the cipher text to the decryption system. This log file indicates the decryption system, the sequence of functions that were executed. The log file is stored as 7ABBCACB. The numeral 7 denotes number of rounds each data set gets encrypted. The same process continues till the plain text ceases. In the base algorithm, the key is composed of data as well as time. That is the key takes the form A5B3C.The user has to enter the character A then at the 5th time unit (time, time unit, character must be known to the sender and receiver) the user has to enter the character 'B' then at the 3rd second (assume the time unit is taken as seconds) 'C' must be entered. To achieve the time inputs real time systems are preferred.

3.2. DECRYPTION ALGORITHM OF BASE SYSTEM

Decryption is done in a system, which works as a real time system. Fig.1 depicts the encryption and decryption process of the base algorithm. When the first part of the key is entered into the decryption program, it is checked for validity. A timer is started which waits for the next part of the key at the right time denoted by 64 bits of time indicating keys. For each and every entry of valid key at valid time interval causes the program to proceed with the decryption. When the full key set is given at appropriate intervals, the decryption process finishes giving out decrypted plain text. If the key or time validity is lost the decryption process will not continue and the full key input has to be given from the first. This process makes every attempt of trying a set of key to consume definite time and thereby the time needed for brute force attack is considerably more. In general, the decryption process is done on a standalone computer, which is in a controlled environment. The decryption process is simply the reverse of the encryption process.

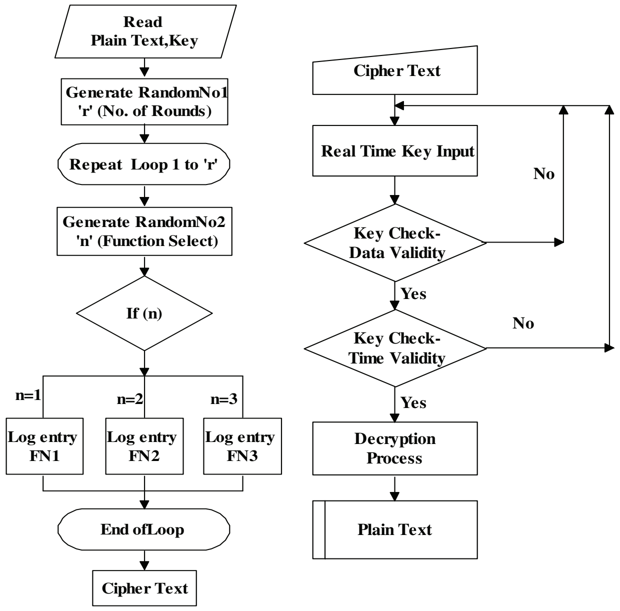


Fig.1. Execution Flow of Base System (Encryption & Decryption)

3.3. ADVANTAGES OF BASE ALGORITHM

1. By adding the time factor in the key, the brute force attack for cryptanalysis can be efficiently protected. The time may take any value and may vary from seconds to many hours thereby increasing the possible choices of cryptanalysis.
2. The size of the key and plain text can easily be altered and flexible for various application requirements.
3. Two random numbers, one for selecting number of rounds and another for selection of functions are used during encryption. Any conventional algorithm can be included as one of the encryption (or decryption) function.
4. These events are properly logged and this log data is also encrypted in the same way as plain text. The decryption system identifies the log data and uses it for decryption.

3.4. DISADVANTAGES

1. In order to achieve the desired result the base algorithm needs to be executed on a real time system. Otherwise a small deviation is allowed to achieve the time factor in the key. A microcontroller can also be used to provide the entire key at the right time.
2. A log file is required to keep track of the number of rounds and the functions that were executed and is also encrypted similar to plain text. This requires additional overhead.

To overcome the above limitations a proposed system is designed that enhances the security without the need for a real time system or a microcontroller in addition to maintaining a log file.

4. CURRENT IMPLEMENTATION

4.1 DESCRIPTION OF THE PROPOSED SCHEME

In this scheme, the key scheduling is carried out by FPGA to generate right key at right time intervals which is then connected to a NIOS – II processor that communicates the keys to the personal computer through JTAG communication. The scheme exercises dynamically varying functions to make the cryptanalysis difficult. Hence, the FPGA implemented with the key scheduler algorithm is used as a hardware key (dongle) for software encryption. This is illustrated in Fig.2.

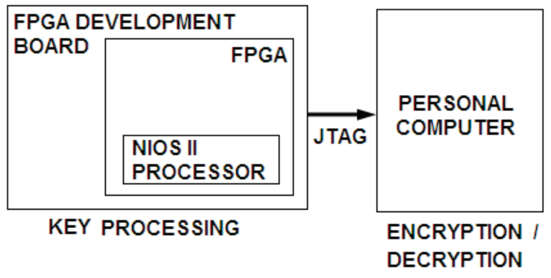


Fig.2. Block Diagram of Proposed Scheme

The proposed algorithm is based on block encryption that encrypts (or decrypts) 64 bits of data using 64 bits of key. A personal computer is used to perform encryption (or decryption). Since, the key scheduler needed for real time operations is implemented on FPGA, it is possible to achieve an accuracy level of nano seconds. However, since the computer's resolution is in the order of micro or milliseconds, the delay can be selected suitably. The key is rotated continuously and four bits are drawn from the key with their concatenated value representing the second dimension of the key, the time. This forms the time based key concept. The function selection is again based on extraction of 8 bit positions where the concatenation of any two bits corresponds to the execution of a different function.

4.2 ENCRYPTION

The following section explains the key scheduler algorithm and its implementation in FPGA. For this a plain text block of 64 bits and key of 64 bits are taken as the inputs. The size of plain text block and key may be changed as it is reprogrammable. The following Fig.3 shows the flowchart of the proposed algorithm. The below operations are performed in a FPGA.

a. Read the key and plaintext.
b. The key is initially passed through a scrambler function to avoid weak keys (A scrambler is a function that eliminates the repetitive pattern in the input data stream and is mainly used to eliminate long sequences consisting of consecutive 1's and 0's). The output is the actual key used for encryption known as 'keyin'.
c. Select four bit positions from the key. Assume the bit positions 12, 24, 36, 48 are chosen but it can also be standardized.
d. The concatenation of these four bits forms the 'timebits'.
e. For every clock, the key is rotated in the anti clockwise and a counter is incremented by one.When the counter value is equal to the 'timebits' the key is ready for the next round. As the key rotates continuously, the four bits drawn from the key varies and hence the number of rotations also varies. Hence, the time taken for processing the key also varies. This forms the time based key concept. Here, key rotation is given as a delay compulsion function (The delay compulsion function is one whose output should progress through the number of clock cycles without getting the results before the designated number of clock cycles. Linear Feedback Shift Register (LFSR) is based Pseudo random number generator output XORed with key with the seed dictated by the key itself can be used as a delay compulsion function).
f. The generated new 'keyin' is passed on to the personal computer for encryption (or decryption).

The following algorithm is implemented on a Personal Computer which performs encryption (or decryption).

a. The personal computer receives the plain text and key of size 64 bits and waits for JTAG port to respond. The key generated from the FPGA is supplied to the personal computer and the system performs encryption (or decryption) with dynamically varying functions.
b. The key is supplied to personal computer since, it serves as a basis for selection of various functions.
c. Now, the rotated key of 64 bits and plain text of 64 bits enters in to round 1. Then 8 bits are extracted from the key as fnbit1, fnbit2, fnbit3 and fnbit4 each of size 2 bits. In conventional algorithms, the sequence of functions is predetermined, whereas in the proposed algorithm, the order of execution of functions is varied based on the selected bits.

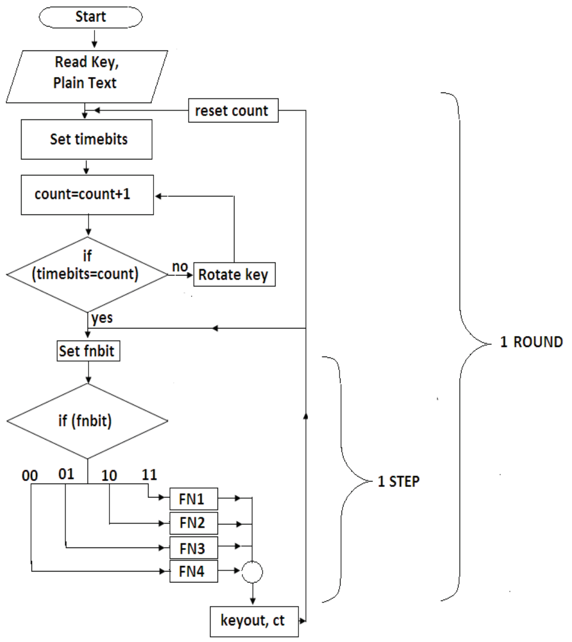


Fig.3. Execution Flow of Proposed Scheme

d. In the proposed algorithm, four functions are used. Function 1 is 'XOR'ing key and plain text bits, Function 2 performs modulo 2 addition, Function 3 is 'XOR'ing the plain text and the key followed by grey code conversion and Function 4 is shuffling the plain text bits.
e. If the fnbit1 is '00' Function 1 is selected and if the fnbit1 is '01' Function 2 is selected and so on. This is step-1 in the round 1.
f. After this, the result of step-1 is given to step-2. In step-2, based on fnbit2, any one of the four functions is selected. The result of step-2 is given to step-3 and goes on.
g. Totally 4 steps are provided in a round based on fnbit1, 2, 3 and 4.
h. After this, the new 'key in' and the output of round one is given to round two. And in round two, again the four bits from locations 12,24,36,48 are taken out from the new rotated key and key is rotated by that value.
i. This procedure continues for eight rounds.

4.3 DECRYPTION

The decryption process is merely the inverse of the encryption process and is same as the previous scheme except that the NIOS II processor is used as an intermediary. To perform decryption the intended receiver must know three things.

1. the secret key
2. standardized bit positions for deciding number of rotations say 12, 24, 36, 48 (shifting is based on concatenation of bits in the positions)
3. 8 bit positions correspond to function selection (2 bit positions for each function selection)

To perform decryption the last round key and the cipher text comes out of the last round are used. Since all the functions are reversible it is possible to go back and get the plain text. Since the intruder is not aware of the above three things, it is very difficult to locate the key through brute force attack. In this scheme the key scheduling is done by FPGA and the function selection and encryption (or decryption) is carried out by the computer. In this scheme NIOS – II processor serves as a communication medium between the keys and the personal computer through JTAG.

4.4 ANALYSIS OF THE PROPOSED ALGORITHM

In this algorithm, the following issues are analyzed for proper functioning.

1. In the implemented system, 4 bits are taken from the said positions. Hence, the maximum number realized is 16 and hence maximum delay achieved is (16 * 4 ns =) 64 ns. However, if the delay needs to be increased, more number of bits can be used. If 10 bits are picked out 2 10 combinations are possible, hence (1024 * 4 ns =) 4096 ns delay is realized.
2. The counter size (number of bits) is a parameter to design. It is selected based on the number of bits selected in previous step. As the counter bits increases, number of transitions increase and dynamic power consumption increases. Hence, if four bits are selected in the previous step, maximum value is 15 and hence the counter should count up to 17.
3. The clock's time period can be altered. Instead of 4 ns clock, higher time period clocks can be selected which increases the delay.
4. It is possible to dynamically vary the execution of functions based on the two bits drawn from the key.

4.5 ADVANTAGES OF PROPOSED SYSTEM

1. The sequence of operations to be carried out for each round is unpredictable.
2. The encryption (or decryption) process is carried out in a computer and the sub key generation is done by FPGA. Since the amount of data to be transferred is comparatively larger than the key, the data handling is done by the computer and the key is generated by FPGA. Hence the FPGA can be suitably connected to the computer when needed based on security constraints and thus it serves as a hardware key (dongle) for data encryption (or decryption).
3. The proposed system requires no real time environment.

4.6 DISADVANTAGES

An additional requirement of NIOS II Processor inside the FPGA may be considered at times as it consumes more area and power. It is used only for JTAG communication.

4.7 ISSUES OF PROPOSED ALGORITHM

In this algorithm, the following issues are analyzed for proper functioning.

1. In the implemented system, 4 bits are taken from the said positions. Hence, the maximum number realized is 16 and hence maximum delay achieved is (16 * 4 ns =) 64 ns. However, if the delay needs to be increased, more number of bits can be used. If 10 bits are picked out 2 10 combinations are possible, hence (1024 * 4 ns =) 4096 ns delay is realized.
2. The counter size (number of bits) is a parameter to design. It is selected based on the number of bits selected in previous step. As the counter bits increases, number of transitions increase and dynamic power consumption increases. Hence, if four bits are selected in the previous step, maximum value is 15 and hence the counter should count up to 17.
3. The clock's time period can be altered. Instead of 4 ns clock, higher time period clocks can be selected which increases the delay.
4. It is possible to dynamically vary the execution of functions based on the two bits drawn from the key.

5. IMPLEMENTATION RESULTS

The following Fig.4 shows the partial program segment of one round of the proposed algorithm.

```
always@(posedge clk)
if(!start) keyin=keyinput;
else
begin
timebits={keyin[12],keyin[24],keyin[36],keyin[48]};
keyin[63:0]={keyin[62:0],keyin[63]};
count=count+1'b1;
if (count==timebits) keyout=keyin;
end
```

Fig.4. Partial Code Segment of key scheduler

Fig. 5 shows the RTL view of the key scheduler. The first module performs the key scheduling and the second module is the NIOS II processor which communicates the key to the personal computer containing the 'C' code through JTAG.

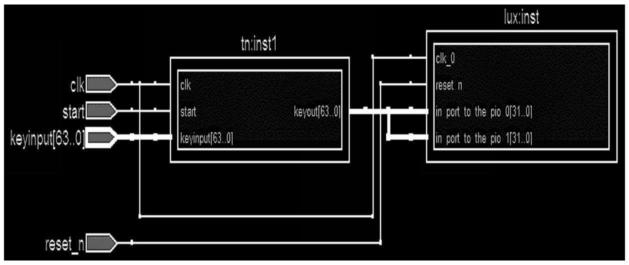


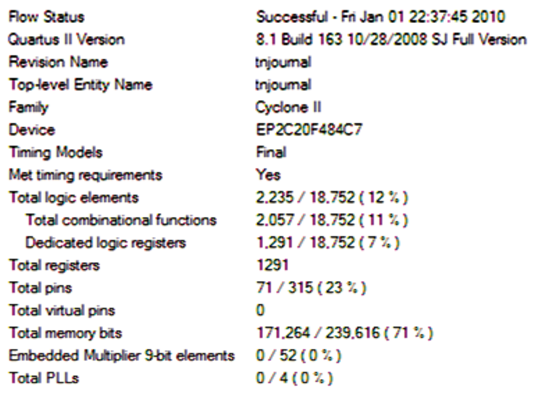
Fig.5. Execution Flow of Base System

The proposed scheme is implemented on an Altera Cyclone II (EP2C20F484C7) device as shown in the compilation summary in Fig. 6. The system consumes 2,057 combinational functions and 1,291 dedicated logic registers.

The following figure, Fig.7 shows the FPGA chip editor view of proposed scheme.

6. CRYPTANALYSIS

For comparing the strength of this algorithm, a mathematical analysis is made with conventional algorithm. For ease of presentation, the following assumptions are made. Assume the key size is 10 bits and there are, totally 8 rounds and each round uses 4 reversible functions.



Flow Status	Successful - Fri Jan 01 22:37:45 2010
Quartus II Version	8.1 Build 163 10/28/2008 SJ Full Version
Revision Name	tnjournal
Top-level Entity Name	tnjournal
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	2,235 / 18,752 (12 %)
Total combinational functions	2,057 / 18,752 (11 %)
Dedicated logic registers	1,291 / 18,752 (7 %)
Total registers	1291
Total pins	71 / 315 (23 %)
Total virtual pins	0
Total memory bits	171,264 / 239,616 (71 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Fig.6.Compilation report of Proposed System

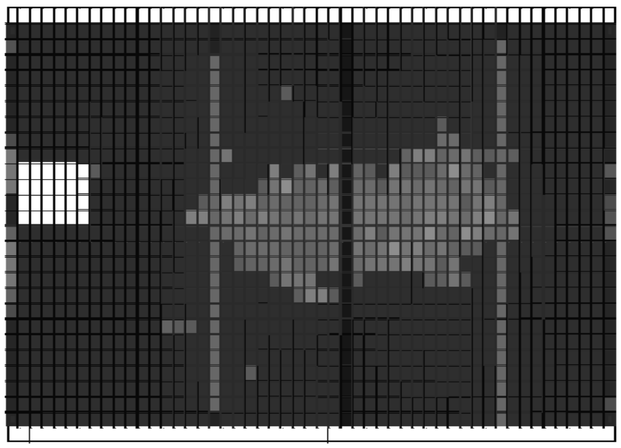


Fig.7. FPGA Chip Editor View of Proposed Scheme

In conventional cryptographic algorithms, with fixed set of equations the brute force attack requires 1024 trials to find the correct key. If the cryptanalysis is done in a FPGA, it is achieved within few nanoseconds. But for the same 10 bit key, in the proposed algorithm, every equation can have any one of the 4 choices. This results in 256 more combinations which tend to increase the level of diffusion and confusion in the cipher text. In addition to this, each function takes time for preparing the key. For example, if 'timebits' is '1010', then the actual key is rotated by 10 bits and time is consumed for 10 counts. If the clock is of 4 ns, then for 10 counts, 40 ns are consumed. This time is again selected from the key input as described earlier and hence, the cryptanalysis takes more time than the conventional algorithm.

7. CONCLUSION

In this paper, a novel encryption algorithm and its implementation on FPGA is presented. The FPGA with the key scheduler algorithm is used as a hardware key for software encryption. The approach with temporal key distribution is proposed to strengthen the security level encryption algorithms against Brute force attack. This approach impedes the fact that, cryptanalysis using brute force attack purely depends on the speed of the system used for cryptanalysis. The base algorithm needs real time system and manual key entry for entering keys at correct time and the minimum allowable time unit an intruder can try is seconds. Also an additional log file is needed for back tracking the operations due to random selection of functions and rounds. The scheme doesn't require data transfer between computer and FPGA (since FPGAs are capable of handling minimum amount of data, in case of bulk processing, data transfer between FPGA and computer is carried out each time which results in pay overhead) since it uses FPGA for key management, computer for performing encryption (or decryption) and NIOS II processor as a communication medium between computer and FPGA. Due to security reasons the scheme uses FPGA as hardware key (dongle) and can be inserted when needed. The proposed scheme comparatively increases the possible number of combinations to try out for cryptanalysis and each combination consumes more time since the basic unit of time is nano seconds. The process of encryption does not consume time and is similar to the existing conventional algorithms and the decryption alone consumes time – a minimum prescribed time for legitimate users and infinite for illegitimate users.

REFERENCES

[1] Omar Elkeelany. 2008. Design and Implementation of Various Models of RC5-192 Embedded Information Security Algorithm. International Journal of Applied Mathematics and Informatics Vol. 2.

[2] Rouvroy.G, Standaert F.X, Quisquater J.J, Legat J.D. 2003. "Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-DES", Proceedings of FPL 2003, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2778, pp. 181-193.

[3] Peng Gong au, Feng-jiao Qiu & Meng Liu, 2004, "A new algorithm based on DES and ECC for CSCW", The 8th International Conference on Computer Supported Cooperative Work in Design, Proceedings, Vol. 1, pp.481–486.

[4] Mukesh Kumar Singh, 2004, "Matrix based asymmetric bulk encryption algorithm", Information Assurance Workshop, Proceedings from the Fifth Annual IEEE SMC, pp. 161 – 167.

[5] Arafat S M, 2005, "An encryption algorithm based on alphabetic trees", The 3rd ACS/IEEE International Conference on Computer Systems and applications, pp.92.

[6] Biham.E and Biryukov.A, 1994, "How to strengthen DES using existing hardware", Advance in Cryptology ASIACRYPT '94, Springer-Verlag.

[7] Simson Garfinkel, Alan Schwartz, Gene Spafford, 2003. Practical Unix & Internet Security**,** 3rd Edition, O'Reilly Publication, ISBN 0-596-00323-4.

[8] Diffie.W, Hellman.M, 1977, “Exhaustive Cryptanalysis of the NBS Data Encryption Standard”, Computer, Vol. 10, pp. 74-84.

[9] Lakshmi .B, Prabakar T.N, Kirubakaran .E, 2008, “Real time cryptography with dual key encryption”, IEEE International Conference on Computing, Communication and Networking, ISBN 9781-4244-3595-1, pp. 1 – 4.

[10] Meier. W, 1993, “On the security of the IDEA block cipher”, Advances in Cryptology – Eurocrypt’93: Workshop on the Theory and Application of Cryptographic Techniques, Proceedings, Lecture Notes in Computer Science, Springer-Verlag, Vol. 765, pp.371–385.

[11] Biham.E, Biryukov.A, and Shamir. A , 1999, “ Miss-in-the-middle attacks on IDEA and Khufru”, Fast Software Encryption: 6th International Workshop, FSE’99,Proceedings, Lecture Notes in Computer Science, Springer-Verlag, Vol.1636, pp.124–138.

[12] Elbirt.J, Chetwynd.B, Yip.W and Paar. C, 2000, “An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists”, AES Candidate Conference, pp.13–27.

[13] Ted Huffmire, Timothy Sherwood, Ryan Kastner, Timothy Levin, “Enforcing memory policy specifications in reconfigurable hardware”, Computers & Security, Vol. 27, pp. 97 – 215, Published by Elsevier Ltd.