VOL. 10, NO. 10, JUNE 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

BLOWFISH ENCRYPTION ALGORITHM FOR INFORMATION SECURITY

Saikumar Manku[1] and K. Vasanth[2]
[1]VLSI Design, Sathyabama University, Chennai, India
[2]Department of Electrical and Electronics Engineering, Sathyabama University, Chennai, India
E-Mail: anandsai.kumar53@gmail.com

ABSTRACT

In this paper, a Blowfish encryption algorithm for information secruity is designed and analyzed. The work is done for networking and communication application for enhanced network security and defence applications. In the proposed Blowfish algorithm reduce rounds of algorithm and proposed single blowfish round. The design simulation is done by Xilinx ISE software using the language of VHDL. Cryptography is the one of the main categories of computer security that converts information from its normal form into an unreadable form. its ability to secure the protected data against attacks and its speed and efficiency in doing so.

Keywords: blowfish, defence, xilinx software, cryptography.

INTRODUCTION

Blowfish is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Blowfish is unpatented and license-free, and is available free for all uses. Blowfish Algorithm is a Feistel Network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors.

Each line - 32 bits.Algorithm keeps two sub-key arrays: The 18-entry P-array four 256-entry S-boxes. S-boxes accept 8-bit input Produce 32-bit output. One entry of P-array is used every round. After final round, each half of data block is XORed with one of the two remaining unused P-entries. The blowfish algorithm Manipulates data in large blocks

Has a 64-bit block size. It has a scalable key, from 32 bits to at least 256 bits. It Uses simple operations that are efficient on microprocessors.e.g., exclusive-or, addition, table lookup, modular- multiplication. It does not use variable-length shifts or bit-wise permutations, or conditional jumps. Employs precomputable subkeys.

On large-memory systems, these subkeys can be precomputed for faster operation. Not precomputing the subkeys will result in slower operation, but it should still be possible to encrypt data without any precomputations. Consists of a variable number of iterations. For applications with a small key size, the trade-off between the complexity of a brute-force attack and a differential attack make a large number of iterations superfluous. Hence, it should be possible to reduce the number of iterations with no loss of security (beyond that of the reduced key size).

BLOWSIH ENCRYPTION

Basically, Blowfish encryption algorithm is requires 32 bit microprocessor at a rate of one byte for every 26 clock cycles. Blowfish contains 16 rounds. Each round consists of XOR operation and a function. Each round consists of key expansion and data encryption. Key expansion generally used for generating intial contents of one arry and data encryption uses a 16 round feiestek network methods. Fig1 shows how blowfish algorithm works. plain text and key are the inputs of this algorithm.64 bit palin text taken is divided into two 32 bits data and at each round the given key is expanded and stored in 18 p-arry and gives 32 bit key as input and XOR ed with previous round data.

Then, for i = 1 to 14:

xL = xL XOR Pi

xR = F(xL) XOR xR

Swap xL and xR

After the sixteenth round, swap xL and xR again to undo the last swap.

Then, xR = xR XOR P15 and xL = xL XOR P16.

Finally, recombine xL and xR to get the ciphertext

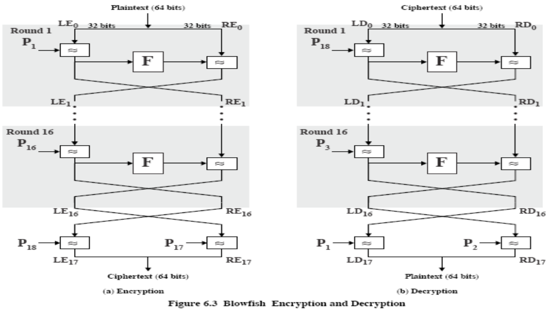


Figure-1. Blowfish encryption and decryption algorithm.

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order.

Generating the Subkeys

Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consist of the hexadecimal digits of pi (less the initial 3): P1 =0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.

XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cyclethrough the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a64-bit key, then AA, AAA, etc., are equivalent keys. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2). Replace P1 and P2 with the output of step (3).

Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys. Replace P3 and P4 with the output of step (5). Continue the process, replacing all entries of the P array, and then all four S-boxes. in orders, with the output of the continuously changing Blowfish algorithm. In total, 521 iterations are required to generate all required subkeys. Store the subkeys rather than execute this derivation process multiple times.

The subkey generation process is designed to preserve the entire entropy of the key and to distribute that entropy uniformly throughout the subkeys. It is also designed to distribute the set of allowed subkeys randomly throughout the domain of possible subkeys. The digits of pi were chosen as the initial subkey table for two reasons: because it is a random sequence not related to the algorithm, and because it could either be stored as part of the algorithm or derived when needed. But if the initial string is non-random in any way (for example, ASCII text with the high bit of every byte a 0), this nonrandomness will propagate throughout the algorithm. In the subkey generation process, the subkeys change slightly with every pair of subkeys generated. This is primarily to protect against any attacked of the subkey generation process that exploit the fixed and known subkeys. It also reduces storage requirements.

The 448 limit on the key size ensures that the every bit of every subkey depends on every bit of the key.

Proposed system of Blowfish

In the proposed system of blowfish alogorithm reduced the rounds of blowfish algorithm and in the algorithm each single round is introduced new modified. See Figure-2 is drawn in below. In the blowfish algorithm there will be 64 bits then the bits are separate into 32bits and there will be four s-boxes. Each s-box contains 32bits. Now design the algorithm like two s-boxes connecting with Xor as like same other two 2 s-boxes connected with Xor and then from the two Xor added then from there get key plain text.

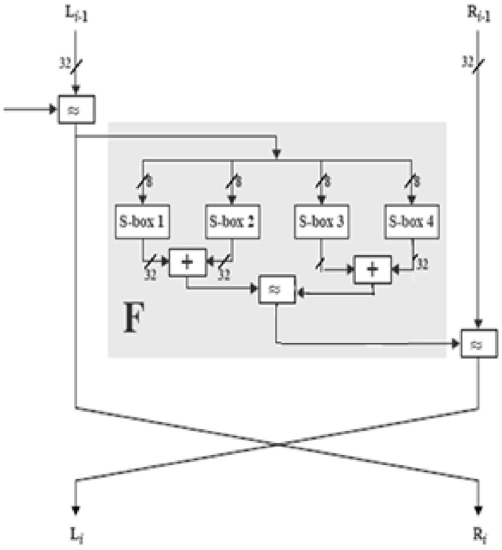
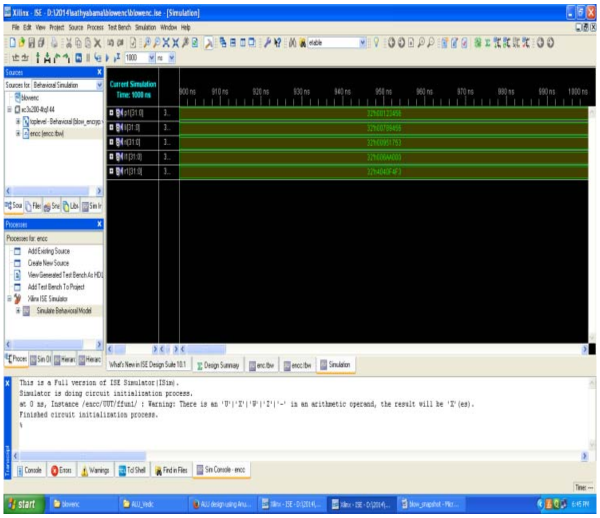


Figure-2. Proposed of blowfish single round.

RESULTS FOR BLOWFISH ALGORITHM

The simulated results for these designs are shown in the Figure-3. In Figure-3 is blowfish encryption algorithm key plain text 1, 2 given value some value and then output of the encryption will get in here then these values note side. In Figure-4 is blowfish decryption. Algorithm decrypting is nothing but reverse order of encryption of bowfish algorithm and in this decryption process from the encryption outputs given in inputs of the decryption and then output of the decryption will come same of encryption inputs.


Xilinx - ISE - D:\2014\sathyabama\blowenc\blowenc.ise - [Simulation]

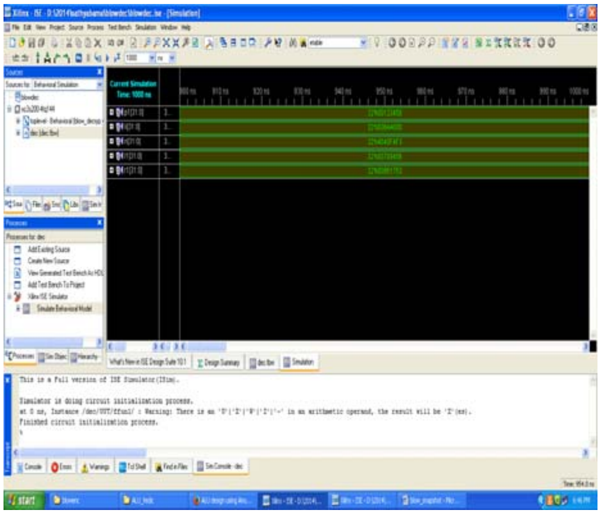
Sources for: Behavioral Simulation

Current Simulation Time: 1000 ns

p1[31:0] 3... 32'h00123456
ll[31:0] 3... 32'h00789456
rl[31:0] 3... 32'h00951753
ll1[31:0] 3... 32'h006AA000
rl1[31:0] 3... 32'h4040F4F3

```
This is a Full version of ISE Simulator(ISim).
Simulator is doing circuit initialization process.
at 0 ns, Instance /encc/UUT/ffun1/ : Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
Finished circuit initialization process.
%
```

Figure-3. Simulated returns loss for blowfish encryption.



```
This is a Full version of ISE Simulator(ISim).

Simulator is doing circuit initialization process.
at 0 ns, Instance /dec/UUT/ffun1/ : Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
Finished circuit initialization process.
```

Figure-4. Blowfish decryption.

CONCLUSIONS

The present simulation result shows that the encryption and decryption is done using blowfish algorithm. Here the algorithm is modified so it provides great security thus no one in between sender and receiver will hack the data.

REFERENCES

[1] Alan G. Konheim. 2007. COMPUTER SECURITY AND CRYPTOGRAPHY. By John Wiley and Sons, Inc.

[2] Alfred J.M., Paul V. C. and Scott A. V. 2001. Handbook of Applied Cryptography. Fifth Addition.

[3] Bruce Schneier. 1996. Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C. Wiley Computer Publishing, John Wiley and Sons, Inc.

[4] B. Schneier. 1994. Applied Cryptography, John Wiley and Sons, New York.

[5] B. Schneier. 1994. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish) Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag. pp. 191-204.

[6] Oppersmith Don. 1994. The data encryption standard (DES) and its strength against attacks. IBM Journal of Research and Development. 38(3): 243-250.

[7] National Institute of Standards and Technology. 1979. FIPS-46: Data Encryption Standard (DES). Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999, available at http://csrc.nist7aznml;'.gov/publications/fips/fips46-3/fips46-3.pdf.

[8] Hala Bahjat AbdulWahab1, Abdul Monem S. Rahma. 2009. Proposed New Quantum Cryptography System Using Quantum Description techniques for Generated Curves. The 2009 International conference on security and management, SAM2009, July 13-16 2009, Las Vegas, USA, SAM.

[9] Henk C.A. van Tilborg, Eindhoven. 2012. Encyclopedia of Cryptography and Security. 2005, Springer Science+Business Media, Inc.