Oct - Dec © 2014 Transactions; 1(4): 149-156

International Journal of Research and Applications

ISSN (online): 2349-0020 http://www.ijraonline.com/

Research Article

FPGA Implementation DIFFIE-HELLMAN key exchange algorithm using Symmetric Encryption Algorithm

B. Keerthi Sudha [1] and P.Rajani [2]

Corresponding Author:
panthaganti189@gmail.com

DOI:
http://dx.doi.org/
10.17812/IJRA.1.4(30)2014

Manuscript:
Received: 16th Sep, 2014
Accepted: 25th Nov, 2014
Published: 1st Dec, 2014

ABSTRACT

Zero-knowledge proof (ZKP) plays an important role in authentication without revealing secret information. Diffie–Hellman (D-H) key exchange algorithm was developed to exchange secret keys through unprotected channels. Previously we have Diffie-hellmen key exchange algorithm. It has some security attacks like man in the middle attack to overcome this attack by using zero knowledge proof concepts. In Diffie Hellman algorithm we had generated one key. That key we have to use in des encryption and decryption .this paper is implemented in Xilinx 13.2 version and verified using Spartan 3e kit.

Keywords: Diffie-hellmen key exchange, des encryption, decryption.

[1] M.Tech (Pursuing) and [2] Assistant Professor
[1 2] Department of ECE, Vaagdevi College of Engineering (Autonomous),
Affiliated to Jawarlal Nehru Technological University, Bollikuntta, Warangal - 506 005.

IJRA - Year of 2014 Transactions:
Month: October - December
Volume – 1, Issue – 4, Page No's:149-156
Subject Stream: Electronics

Paper Communication: Author Direct

Paper Reference Id: IJRA-2014: 1(4)149-156

International Journal of Research and Applications
Oct - Dec © 2014 Transactions; 1(4): 149-156
ISSN (online): 2349-0020
http://ijraonline.com
ELECTONICS RESEARCH ARTICLE

FPGA Implementation DIFFIE-HELLMAN key exchange algorithm using Symmetric Encryption Algorithm

B. Keerthi Sudha [1] and P.Rajani [2]
[1] M.Tech(Pursuing) and [2] Assistant Professor
[1 2] Department of ECE, Vaagdevi College of Engineering (Autonomous),
Affiliated to Jawarlal Nehru Technological University, Bollikuntta, Warangal - 506 005.
[1] keerthisudha.bhandaru@gmail.com and [2] panthaganti189@gmail.com

ABSTRACT

Zero-knowledge proof (ZKP) plays an important role in authentication without revealing secret information. Diffie–Hellman (D-H) key exchange algorithm was developed to exchange secret keys through unprotected channels. Previously we have Diffie-hellmen key exchange algorithm. It has some security attacks like man in the middle attack to overcome this attack by using zero knowledge proof concepts. In Diffie Hellman algorithm we had generated one key. That key we have to use in des encryption and decryption .this paper is implemented in Xilinx 13.2 version and verified using Spartan 3e kit.

Keywords: Diffie-hellmen key exchange, des encryption, decryption.

1. INTRODUCTION

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War I and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread. Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances, e.g., improvements in integer factorization algorithms, and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that provably cannot be broken even with unlimited computing power an example is the pad but these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

Encryption is the process by which meaningful data, such as a file, is transformed into meaningless data that cannot be read without decrypting the data back into its meaningful form. In the vast majority of applications in which encryption is used, only one or very few people (or computers) have the knowledge and ability to decrypt the Encrypted data and make it useful again. An encrypted file or data stream looks like a jumble of random letters, numbers, and other characters, and it is impossible to make sense of encrypted data without

transforming it back into its original form. To perform this transformation, we need to know the cipher that was used to encrypt the data. A cipher is a set of logical instructions that can be followed in the same way every time to produce the same result. During the process of encryption, the instructions in the cipher are performed on the data to be encrypted. These steps change the readable data, or plaintext, into its unreadable, encrypted form, or cipher text. Some ciphers also need a key in order to work on data. A key is a small piece of data that is used to customize each cipher. Each time a particular cipher is used, the key is changed. This allows a cipher to be used more than once without compromising its integrity. If a particular cipher was widely used and could be decrypted without a key, it wouldn't be secure. Anyone would be able to decrypt any data encrypted with this cipher. If keys didn't exist, a person would have to write a new cipher every time he or she wanted to encrypt a piece of data! It is very easy to generate a key, but very difficult to write a secure, effective cipher

Encryption has been in use for thousands of years. The earliest known encryption was found engraved on Egyptian monuments from around 2500 B.C. Hebrew scholars used simple alphabetic substitutions, in which one letter of the alphabet stands for one other letter, with no letters standing for more than one and no letters being omitted. The ancient Greeks used encryption to transmit military messages. Encryption was used heavily during World War II to obscure radio transmissions and telegrams

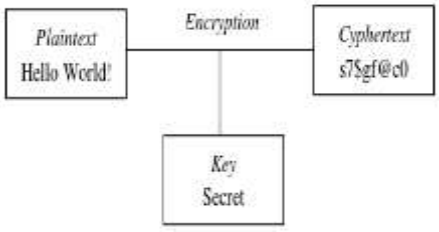


Figure: 1

Cryptography is an algorithmic process of converting a plain text or clear text message to a cipher text or cipher message based on an algorithm that both the sender and receiver know, so that the cipher text message can be returned to its original, plain text form. In its cipher form, a message cannot be read by anyone but the intended receiver. The act of converting a plain text message to its cipher text form is called enciphering. Reversing that act (i.e., cipher text form to plain text message) is deciphering. Enciphering and deciphering are more commonly referred to as encryption and decryption, respectively. There are a number of algorithms for performing encryption and decryption, but comparatively few such algorithms have stood the test of time. The most successful algorithms use a key. A key is simply a parameter to the algorithm that allows the encryption and decryption process to occur. There are many modern key-based cryptographic techniques. These are divided into two classes: symmetric and asymmetric (also called public/private) key cryptography. In symmetric key cryptography, the same key is used for both encryption and decryption. In asymmetric key cryptography, one key is used for encryption and another, mathematically related key, is used for decryption

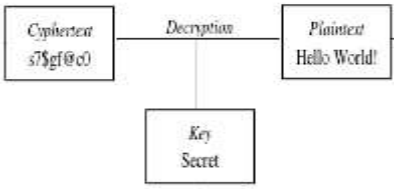


Figure: 2

2. Diffie Hellman Key Exchange

It is a specific method of exchanging cryptographic keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. The scheme was first published by Whitfield

Diffie and Martin Hellman in 1976, although it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, by James H. Ellis, Clifford Cocks and Malcolm J. Williamson but was kept classified.[citation needed] In 2002, Hellman suggested the algorithm be called Diffie–Hellman–Merkle key exchange in recognition of Markel's contribution to the invention of public-key cryptography (Hellman, 2002).Although Diffie–Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).The method was followed shortly afterwards by RSA, an implementation of public key cryptography using asymmetric algorithms.

The system...has since become known as Diffie–Hellman key exchange. While that system was first described in a paper by Diffie and me, it is a public key distribution system, a concept developed by Merkle, and hence should be called 'Diffie–Hellman–Merkle key exchange' if names are to be associated with it. I hope this small pulpit might help in that endeavour to recognize Markel's equal contribution to the invention of public key cryptography.

A zero-knowledge proof (ZKP) is a proof of some statement which reveals nothing other than the veracity of the statement. The word "proof" here is not used in the traditional mathematical sense. Rather, a "proof", or equivalently a "proof system", is an interactive protocol by which one party (called the prover) wishes to convince another party (called the verifier) that a given statement is true. In ZKP, the prover proves that he/she knows a secret without revealing it. Researches in zero-knowledge proofs has been motivated by authentication systems where one party wants to prove its identity to a second party via some secret information (such as a password) but doesn't want the second party to learn anything about this secret. This is called a "zero-knowledge proof of knowledge". However, a password is typically too small or insufficiently random to be used in many schemes for zero-knowledge proofs of knowledge. A zero-knowledge password proof is a special kind of zero- knowledge proof of knowledge that addresses the limited size of passwords. One of the most fascinating uses of zero-knowledge proofs within cryptographic protocols is to enforce honest behaviour while maintaining privacy. Roughly, the idea is to enforce a user to prove, using a zero-knowledge proof, that its behaviour is correct according to the protocol. Because of soundness, we know that the user must really act honestly in order to be able to provide a valid proof. Because of zero knowledge, we know that the user does not compromise the privacy of its secrets in the process of providing the proof.

Diffie-Hellman key exchange algorithm was invented in 1976during collaboration between Whitfield Diffie and Martin Hellman and was the first practical method for establishing a shared secret between two parties (Alice and Bob) over an unprotected communications channel. The protocol uses the multiplicative group of integers modulo p <Zp*, x>, where p is a prime number. That simply means that the integer's between1 and p−1 are used with normal multiplication, exponentiation and division, except that after each operation the result keeps only the remainder after dividing by p. The two parties (Alice and Bob) need to choose two numbers p and g; where p (modulo) is a prime number and the second number g is a primitive root of order (p-1) in the group <Zp*,x> called the generator. The two numbers are public and can be sent through the Internet

1. Alice chooses a large random number x, such that $0<x<p$ and calculate $R_1 = g^x \bmod p$.
2. Bob chooses another large random number y, such that $0<y<p$ and calculate $R_2 = g^y \bmod p$.
3. Alice sends R_1 to Bob.
4. Bob sends R_2 to Alice.
5. Alice computes $K_{Alice} = (R_2)^x \bmod p$.
6. Bob computes $K_{Bob} = (R_1)^y \bmod p$.

Both Alice and Bob have arrived at the same key value;

$K_{Alice} = (R_2)^x \bmod p = (g^y \bmod p)^x \bmod p = g^{xy} \bmod p.$

$K_{Bob} = (R_1)^y \bmod p = (g^x \bmod p)^y \bmod p = g^{yx} \bmod p.$

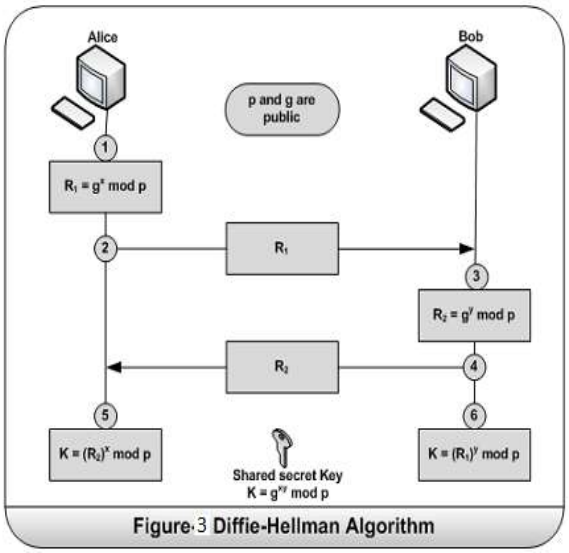


Figure-3 Diffie-Hellman Algorithm

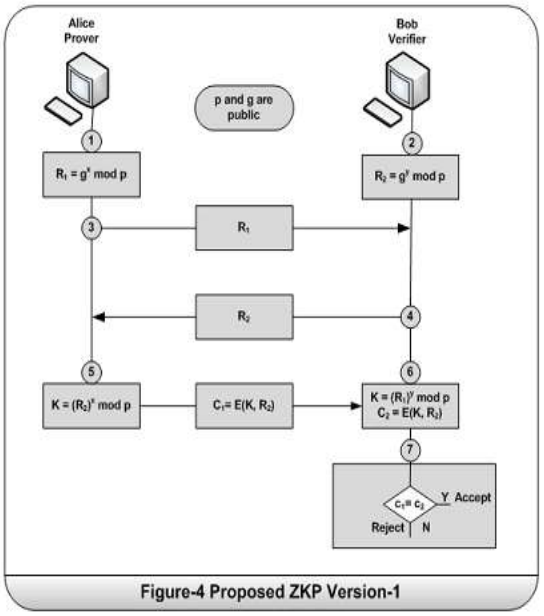


Figure-4 Proposed ZKP Version-1

The proposed ZKP based on D-H key exchange algorithm in the sense that both parties (the prover and the verifier) exchange non secret information and did not revealing secrets to get one identical secret key. This means that the prover can prove to the verifier that he knows the secret. The proposed algorithm developed in two stages; in the first stage we develop a first version based on the basic D-H key exchange algorithm which is vulnerable to man-in-the-middle-attack. The second version has been developed to resists the man-in the-middle attack. The two versions will be describes below

1) Alice (the prover) chooses a large random number x, such that $0<x<p$ and calculate $R_1 = g^x \bmod p$.
2) Bob (the verifier) chooses another large random number y, such that $0<y<p$ and calculate $R_2 = g^y \bmod p$.

3) Alice sends R_1 to Bob.
4) Bob sends R_2 to Alice.
5) Alice (the prover), computes $K_{Alice} = (R_2)^x \bmod p$, and send encrypted R_2 to Bob using K_{Alice} ($C_1 = E(K_{Alice}, R_2)$).
6) Bob computes $K_{Bob} = (R_1)^y \bmod p$, and calculate ($C_2 = E(K_{Bob}, R_2)$).
7) Bob (the verifier) verify ($C_1 = C_2$); if equal then Alice is accepted, otherwise it is rejected.

1) Alice (the prover) chooses a large random number x, such that $0<x<p$ and calculate $R_1 = g^x \bmod p$.
2) Alice sends R_1 to Bob.
3) Bob (the verifier) chooses another large random number y, such that $0<y<p$ and calculate $R_2 = g^y \bmod p$, $K_{Bob} = (R_1)^y \bmod p$, and $C_1 = E(K_{Bob}, R_2)$.
4) Bob sends ($R_2 \mid C_1$) to Alice.
5) Alice, calculates $K_{Alice} = (R_2)^x \bmod p$, decrypt ($R_2' = D(K_{Alice}, C_1)$) and verify ($R_2 = R_2'$). If they matched then she proceeds; otherwise the verifier is dishonest.
6) Alice encrypt ($C_2 = E(K_{Alice}, R_1|R_2)$) and send it to Bob.
7) Bob decrypt C_2 to get R_1' and R_2'
8) Bob verify ($R_1 = R_1'$); if they are equal then Alice is verified (Accepted), otherwise it is a dishonest prover (rejected).

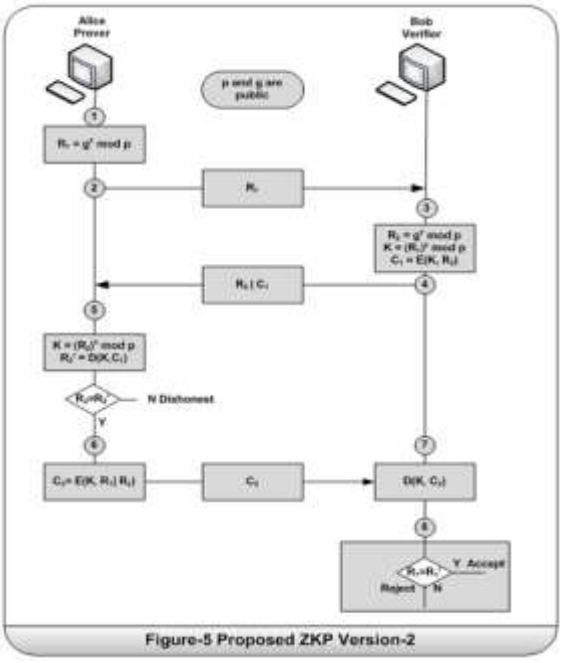


Figure-5 Proposed ZKP Version-2

3. DES

DES was the result of a research project set up by International Business Machines (IBM) Corporation in the late 1960's which resulted in a cipher known as LUCIFER. In the early 1970's it was decided to commercialize LUCIFER and a number of significant changes were introduced. IBM was not the only one involved in these changes as they sought technical advice from the National Security Agency (NSA) (other outside consultants were involved but it is likely that the NSA were the major contributors from a technical point of view). The altered version of LUCIFER was put forward as a proposal for the new national encryption standard requested by the National Bureau of Standards (NBS). It was finally adopted in 1977 as the Data Encryption Standard -DES (FIPS PUB 46) DES (and most of the other major symmetric ciphers) is based on a cipher known astheFeistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a feistel network).As with most encryption schemes; DES expects two inputs - the plaintext to be en-crypted and the secret key. The manner in which the plaintext is accepted, and the key arrangement used for encryption and decryption, both determine the type of cipher it is. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and only operates on 64 bit blocks of data at a time5(be they plaintext or cipher text). The key size used is 56 bits, however a 64 bit (or eight-byte) key is actually input. The least significant bit of each byte is either used for parity (odd for DES) or set arbitrarily and does not increase the security in any way. All blocks are numbered from left to right which makes the eight bit of each byte the parity bit. Once a plain-text message is received to be encrypted, it is arranged into 64 bit blocks required for input. If the number of bits in the message is not evenly divisible by 64, then the last block will be padded. Multiple permutations and substitutions are incorporated throughout in order to increase the difficulty of performing a cryptanalysis on the cipher. However, it is generally accepted that the initial and final permutations offer little or no contribution to the security of DES and in fact some software implementations omit them (although strictly speaking these are not DES as they do not adhere to the standard)

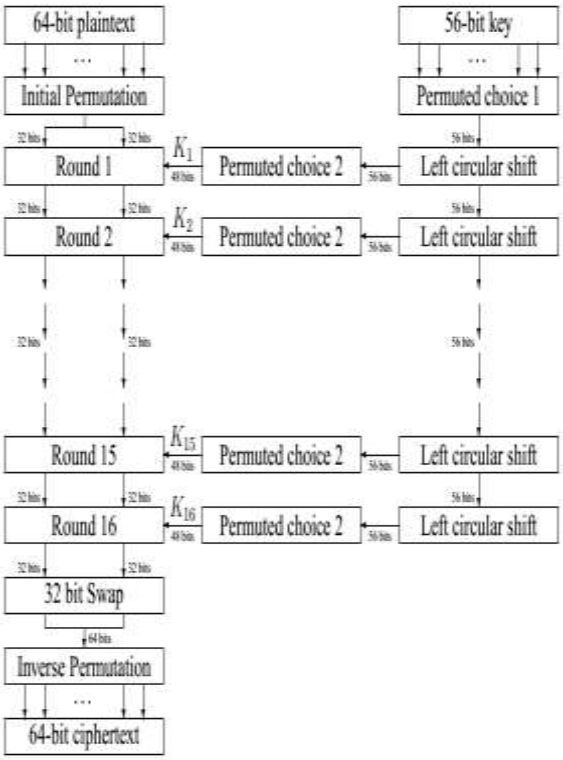


Figure: 6

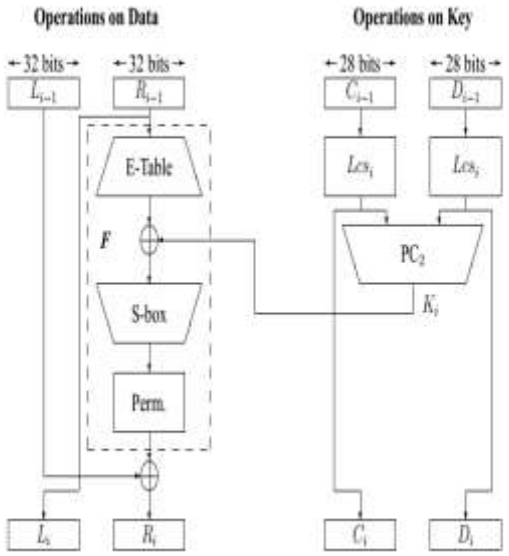


Figure: Round operation

The sequence of events that occur during an encryption operation.DES performs an initial

permutation on the entire 64 bit block of data. It is then split into 2, 32 bit sub-blocks, Li and Ri which are then passed into what is known as around of which there are 16 Each of the rounds are identical and the effects of increasing their number is twofold - the algorithms security is increased and its temporal efficiency decreased. Clearly these are two conflicting outcomes and a compromise must be made. For DES the number chosen was 16, probably to guarantee the elimination of any correlation between the cipher text and either the plaintext or key. At the end of the16th round, the 32 bit Li and RI output quantities are swapped to create what is known as the pre-output. This [R16, L16] concatenation is permuted using a function which is the exact inverse of the initial permutation. The output of this final permutation is the 64 bit cipher text.

As shown in Figure, the 48-bit input word is divided into eight 6-bit words and each 6-bit word fed into a separate S-box. Each S-box produces a 4-bit output. Therefore, the 8 S-boxes together generate a 32-bit output. As you can see, the overall substitution step takes the 48-bit input back to a 32-bit output.

Each of the eight S-boxes consists of a 4 × 16 table lookup for an output 4-bit word. The first and the last bit of the 6-bit input word are decoded into one of 4 rows and the middle 4 bits decoded into one of 16 columns for the table lookup.

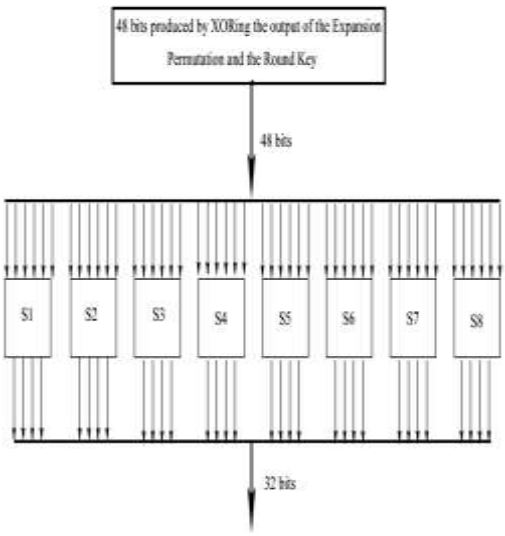


Figure: S Box Block Diagram

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

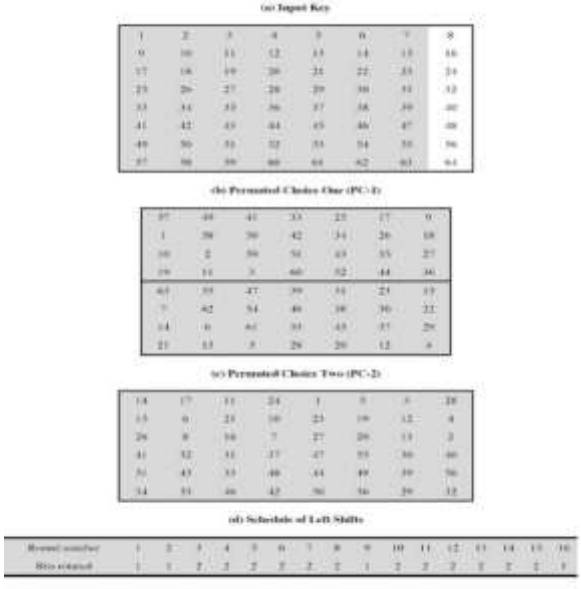
(d) Permutation Function (P)

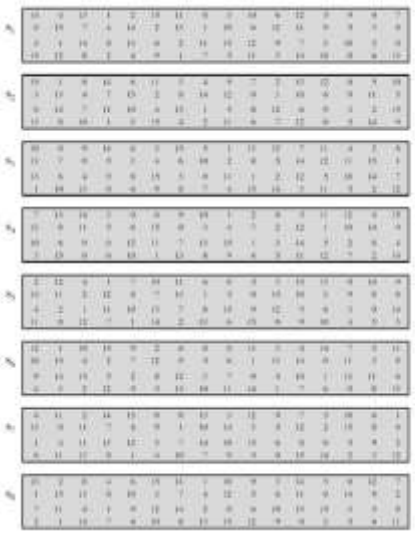
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

The goal of the substitution carried out by an S-box is to enhance diffusion.

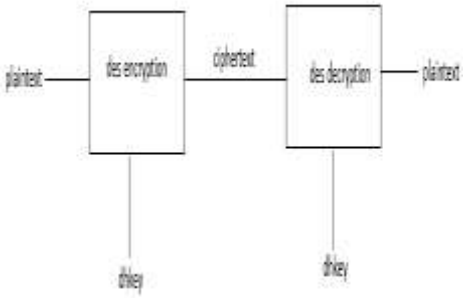
Shown on the below figure are the eight S-boxes, S1 through S8,each S-box being a 4×16 substitution table that is used to convert 6 incoming bits into 4 outgoing bits.

As mentioned earlier, each row of a substitution table is indexed by the two outermost bits of a six-bit block and each column by the remaining inner 4 bit





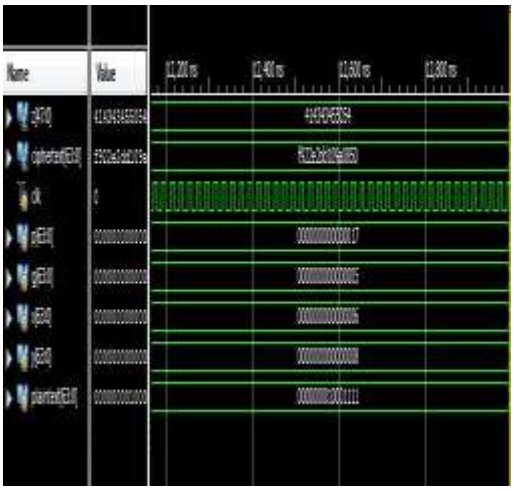
4. PROPOSED OPERATION



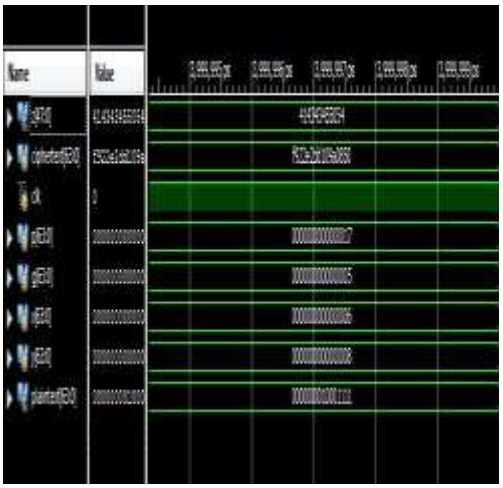
5. SIMULATION RESULTS

In this Work XILINX ISE 13.2 simulator used.

Encryption:



Decryption:



6. Conclusion

The Diffie-Hellman key exchange exploits mathematical properties to produce a common computational result between two (or more) parties wishing to exchange information, without any of them providing all the necessary variables. By agreeing on two variables and providing each other with a computed public key, the resulting secret key will be identical throughout the exchange.

It is, of course, possible to intervene by either masquerading or by sheer brute force, but the first is a common concern — authentication — which must be addressed separately, and the second is, when done right, computationally infeasible. With proper authentication mechanisms, proper prime generation, and true randomness in picking variables, the D-H protocol can be a powerful component in many a security measure. Good implementations include usages in Secure Sockets Layer, Secure Shells, IP Security, and others. Passages on short introductory and background information, description, issues, as well as common usages were produced in this report, and it is hoped that they were all of sufficient clarity. The Shared secret key is generated using diffie-hellman algorithm using authentication to provide more security. That key we are going to use for symmetric encryption algorithm. In this paper we used DES algorithm for communication between two parties.

7. REFERENCES

[1] Back, Amanda, (2009), "The Differ-Hellman Key Exchange", December 2, 2009.

[2] Carts, David A., (2001), "A Review of the DiffieHellman Algorithm and its Use in Secure Internet Protocols", SANS Institute, 2001.

[3] Clausen, Andrew, (2007), "Logical Composition of Zero-Knowledge Proofs", Oct, 2007, http://www.econ.upenn.edu/_clausen.

[4] Endre Bangerter, etal, (2009), "On the Design and Implementation of Efficient Zero- Knowledge Proofs of Knowledge", Proceedings of the 2nd ECRYPT Conference on Software Performance Enhancement for Encryption and Decryption and Cryptographic Compilers (SPEED-CC'09), Berlin, Germany, Oct 2009.

[5] Fischer, Michael J., (2010), "Cryptography and Computer Security", Department of Computer Science, Yale University, Mar 29.

[6] Forouzan, Behrouz A. (2008), "Cryptography and Network Security", McGraw-Hill, Int. Ed. 2008.

[7] Hellman, Martin E., (2002), "An Overview of PublicKey Cryptography", IEEE communications Magazine, May 2002, pp: 42-49.

[8] Kizza, Joseph M, (2010), "Feige-Fiat-Shamir ZKP Scheme Revisited", International Journal Of Computing and ICT Research, Vol. 4, No. 1, June 2010.

[9] Krantz, Steven G., "Zero Knowledge Proofs", AIM Preprint Series, Volume 10-46, July25, 2007.

[10] Maurer Ueli,, "Unifying Zero-Knowledge Proofs of Knowledge", Africacrypt, pp. 272–286, 2009.

[11] Michael Backes and Dominique Unruha, (2009), "Computational Soundness of Symbolic Zero Knowledge Proofs", Journal of Computer Security, Vol. 18, No. 6, pp. 1077-1155, 2010.

[12] Mohr, Austin "A Survey of Zero-Knowledge Proofs with Applications to Cryptography", Southern Illinois University, 2007.

[13] P. Bhattacharya, M. Debbabi and H. Otrok, "Improving the Diffie-Heliman Secure Key Exchange", International Conference on Wireless Networks, Communications & Mobile Computing, 2005.

[14] Simari, Gerardo I. "A Primer on Zero Knowledge Protocols", Technical report, Universidad Nacional del Sur, Buenos aires, argentina, 2002.

[15] Stallings, William (2010), "Cryptography and Network Security", Prentice Hall, 5th Ed. 2010.

[16] Velten, Michael, "Zero-Knowledge the Magic of Cryptography", Saarland University, August, 2006.

[17] Liu Wan, He Daojun, Tan Ming. "FPGA technology and application". Beijing: [2] Tsinghua University Press, 2006.

[18] Zhang Huanguo, Liu Yuzhen. "Cryptography Theory". Wuhan: Wuhan University Press, 2003.

[19] Xu Guanghui, Cheng Dongxu, Huang Ru, "Embedded development and application based on FPGA", Beijing: Publishing House of Electronics Industry, 2006.

[20] Shao Junxiang, He Zhimin. "High-speed implementation of 3DES encryption algorithm based on FPGA". Modern electronic technology, 2004.

[21] "National Institute of Standard and Technology". Data Encryption Standard (DES)[EB/OL].1999.

[22] McLoone, McCanny, J.V. "A high performance FPGA implementation of DES". IEEE Conference on Signal Processing Systems[C]. Washington: IEEE Computer Society Press, 2000: 374-383.

AUTHOR'S BIOGRAPHY

Student Name : B. Keerthi Sudha
Branch : VLSI
Qualification : M.Tech (Pursuing)

Guide Name : P.Rajani
Qualification : M.Tech
Designation : Assistant Professor