IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 9, Issue 6, Ser. I (Nov. - Dec. 2019), PP 15-24
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

FPGA Implementation of Modified Diffie-Hellman Key Exchange Algorithm Using Zero Knowledge Proof

D.Surendra Rao
Associate Professor, Department of Electronics and Communication Engineering, GNITC – Hyderabad

Abstract: *There are networks and entity groupings that require authentications while preserving the privacy of the entity being authenticated. Zero – Knowledge Proof (ZKP) plays a vital role in authentications without revealing secret information. Our proposed work carries criticism of ZKP, and Diffie–Hellman Key Exchange Algorithm (DHKEA). A new ZKP has been proposed based on modifications of the DHKEA. As per the modifications, two versions of the proposed protocols are developed. Verilog HDL is effectively utilized to complete the design of proposed protocols. Results will be verified through simulations and FPGA target board.*

Key words: *Authentication, Networks, Entity groupings Zero-Knowledge Proof, Verilog HDL, Simulations, FPGAs*

Date of Submission: 10-02-2020 Date of Acceptance: 25-02-2020

I. Introduction

It is a common weakness among traditional communication protocols to be vulnerable to impersonation attacks. Every time this sort of protocol is executed, the system degrades because of the threat of an eavesdropper listening in on the communications [1]. Zero-Knowledge Proof protocols presented by Gold wasser, Micali and Rack off, which are an improvement on these situations. The objective of ZKP is, to obtain a system in which it is possible for a prover to convince a verifier of his knowledge of a certain secret without disclosing any intelligence except the validity of his claim.

Diffie-Hellman Key Exchange Algorithm (DHKEA) is having a specific method of exchanging secret keys. It is one of the earliest practical examples of key exchange algorithm implemented within the field of cryptography. The DHKEA allows two parties prover and verifier that have no prior knowledge of each other to jointly establish a secret key over an insecure communication channels. This key can then be used to encrypt subsequent communication using symmetric key ciphers. It is designed specifically to exchange secret key in insecure communication channels. But it is not venerable to the impersonation attacks [2,5].

In our work we are criticizing a ZKP, D-H Key Exchange Algorithm. A new ZKP has been proposed, based on modifications of D-H Key Exchange Algorithm. We developed two versions of the protocols using Verilog HDL and implemented on FPGA target board for physical verification of results.

The organization of the work follows as; in section I we try to describe the basic information and motivation towards the work. In section II we carried out some basic discussions on cryptography, ZKP. In section III we concentrated D-H Key Exchange Algorithm and motivation towards modifications required in proposed algorithm and their protocols [7]. In section IV specifies the design of proposed versions its importance. In section V we conclude our work with simulation results and emulations.

II. Related Work

Cryptography: Cryptography is the idea of storing and transmitting data in a form that, only the authorized parties' can interpret. Confidentiality of network communications, for example, is of great importance for e-commerce, m-commerce and other network applications. Fig.1 gives basic idea behind in cryptography.

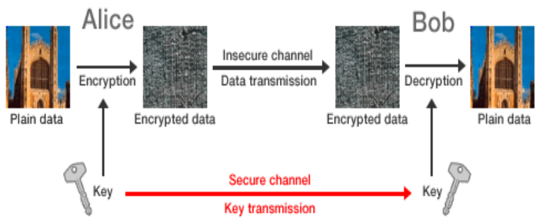

Fig.1 Basic Block Diagram of Cryptography

Types of key algorithms: In cryptography, a cipher is an algorithm for performing encryption or decryption a series of well-defined steps that can be followed as a procedure. The operation of a cipher usually depends on a piece of auxiliary information called key. The encrypting procedure is varied depending on a key, which changes the detailed operation of the algorithm. A key must be selected before using a cipher to encrypt a message. Without knowledge of the key, it should be extremely difficult, if not impossible, to decrypt the resulting cipher text into readable plain text, most modern ciphers can be categorized in several ways;

i.) Symmetric Key Algorithm: The same key is used for both encryption and decryption.
ii.) Asymmetric Key Algorithm: A different key is used for both encryption and decryption.
iii.) Block Ciphers: They works on blocks of symbols usual of a fixed size.
iv.)Stream Ciphers: They work on a continuous stream of symbols.

Zero Knowledge Proof: ZKP model of computation defined as an interactive proof system (P,V), where P is prover and V is verifier. Protocol of (P,V) is , for proving a language membership statement for a language over {0,1}. Let L be a language over {0,1}, for a membership instance x€ L, P and V must share the common input x, proof instance is denoted as $(P,V)_{(X)}$. P and V are linked by a communication channel. Over which they exchange a sequence, called proof transcript a_1, b_1, a_2, b_2 ….a_n, b_n. Proof transcript interleaves prover's transcript and verifier's transcript. Each element a_i, b_i exchanged is bounded by polynomial time in |x|. Upon completing the interaction, the output of the protocol should be of form $(P,V)_{(x)}$ € {Accept, Reject} representing V's acceptance or rejection of P's claim that x € L.

III. DIFFIE-Hellman Key Exchange Algorithm

Asymmetric Encryption of data requires transfer of cryptographic private key. The most challenging part in this type of encryption is the transfer of the encryption key from sender to receiver without anyone intercepting this key in between. This transfer or rather generation on same cryptographic keys at both sides secretively was made possible by the Diffie-Hellman algorithm. The Diffie-Hellman algorithm was developed by Whitfield Diffie and Martin Hellman in 1976. This algorithm was devices not to encrypt the data but to generate same private cryptographic key at both ends so that there is no need to transfer this key from one communication end to another. Though this algorithm is a bit slow but it is the sheer power of this algorithm that makes it so popular in encryption key generation.

[A] The DHKE Algorithm and Protocol:

The protocol uses the multiplicative group of integers modulo p <Zp*,x>, where p is a prime number. That simply means that the integers between 1 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. Figure-2 shows the procedure of the protocol, the steps are as follows:

i. Alice chooses a large random number x, such that 0< x < p and calculate $R_1 = g^x \bmod p$.
ii. Bob chooses another large random number y, such that 0< y< p and calculate $R_2 = g^y \bmod p$.
iii. Alice sends R_1 to Bob.
iv. Bob sends R_2 to Alice.
v. Alice computes $K_{Alice} = (R_2)^x \bmod p$.
vi. Bob computes $K_{Bob} = (R_1)^y \bmod p$. Both Alice and Bob have arrived at the same key value;

$K_{Alice} = (R_2)^x \bmod p = (g^y \bmod p)x \bmod p = g^{xy} \bmod p$.
$K_{Bob} = (R_1)^y \bmod p = (g^x \bmod p)y \bmod p = g^{xy} \bmod p$.

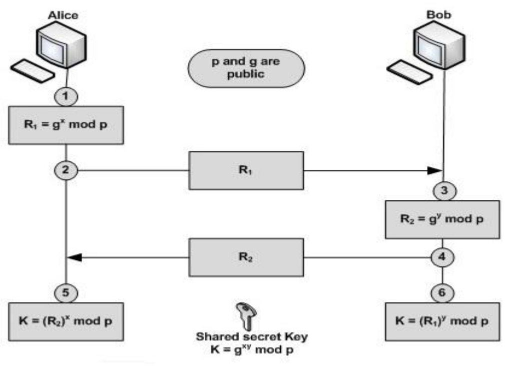

Fig. 2 DH Key Exchange Algorithm

[B] Security of D-H Key Exchange Algorithm:

The protocol is considered secure against eavesdroppers if G and g are chosen properly. The eavesdropper ("Eve") would have to solve the Diffie–Hellman problem to obtain g^{ab}. This is currently considered difficult. An efficient algorithm to solve the discrete logarithm problem would make it easy to compute a or b and solve the Diffie–Hellman problem, making this and many other public key cryptosystems insecure. The order of G should be prime or have a large prime factor to prevent use of the Pohlig–Hellman algorithm to obtain a or b. For this reason, a Sophie Germain prime q is sometimes used to calculate p=2q+1, called a safe prime, since the order of G is then only divisible by 2 and q, g is then sometimes chosen to generate the order q subgroup of G, rather than G, so that the Legendre of g^a never reveals the low order bit of a. If Alice and Bob use random number generators whose outputs are not completely random and can be predicted to some extent, then Eve's task is much easier. The secret integers a and b are discarded at the end of the session. Therefore, D–H key exchange by itself trivially achieves perfect forward secrecy because no long-term private keying material exists to be disclosed.

In the original description, the Diffie–Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack. A person in the middle may establish two distinct Diffie–Hellman key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing the attacker to decrypt (and read or store) then re-encrypt the messages passed between them. The Diffie-Hellman algorithm is susceptible to two attacks.

1. The discrete logarithm attack and
2. The man-in-the-middle attack.

1.) Discrete Logarithm attack

An interceptor (Eve) can intercept u and v. Find a from ($u = g^a \bmod p$);Find b from ($v = g^b \bmod p$); Then she can calculate ($K = g^{ab} \bmod p$). The secret key is not secret anymore. To make Diffie-Hellman safe from the discrete logarithm attack, the following are recommended:

- The prime number p must be very large (more than 300 digits).
- The generator g must be chosen from the group $<Z_P*, a>$.
- The numbers a and b must be large random numbers of at least 100 digits long, and used only once (destroyed after being used). Still, no algorithm for the discrete logarithm problem exists with computational complexity O(x r) for any r; all are infeasible.

2.) Man-in-the-middle attack:

Diffie-Hellman algorithm is vulnerable to the man-in-the-middle attack in which the attacker is able to read and modify all messages between Alice and Bob. As g is not secret, the attacker can easily create his own power of g and send that to Bob. When Bob replies, the attacker intercepts the message and will share his key with Bob. Eve, the interceptors can create two keys; one between herself and Alice, and another between herself and Bob. Figure 3-2 shows the man-in-the-middle attack.

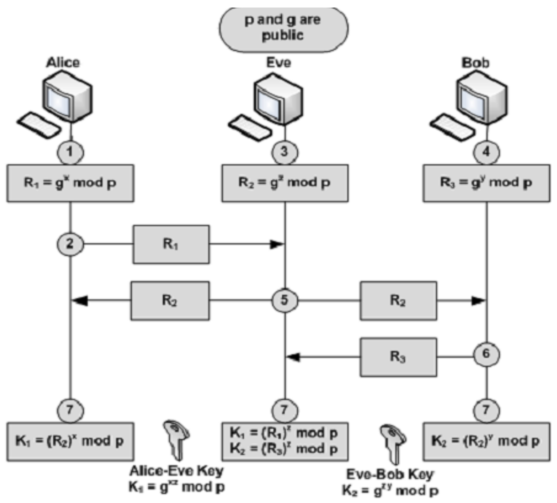

Fig.3 Man in the middle attack

i. Alice chooses x, and calculate $R_1 = g^x$ mod p and sends R 1 to Bob.
ii. Eve, the intruder, intercept R_1 , chooses z, calculates $R_2 = G^z$ mod p, send R_2 to both Alice and Bob.
iii. Bob chooses y, and calculate $R_3 = g^y$ mod p and sends R_3 to Alice. R_3 is intercepted by Eve and never reaches Alice.
iv. Alice and Eve calculate $K_1 = g^{xz}$ mod p, which becomes shared key between them.
v. Eve and Bob calculate $K_2 = g^{zy}$ mod p, which becomes shared key between them.

IV. Proposed Modifications

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. 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.

[A] Algorithm of Proposed ZKP Version-1:

i. Alice (the prover) chooses a large random number x, such that $0< x < p$ and calculate $R_1 = g^x$ mod p.
ii. Bob (the verifier) chooses another large random number y, such that $0< y < p$ and calculate $R_2= g^y$ mod p.
iii. Alice sends R_1 to Bob.
iv. Bob sends R_2 to Alice.
v. Alice (the prover), computes $K_{Alice} = (R_2)^{x}$ mod p, and send encrypted R_2 to Bob using K_{Alice}
vi. (C_1=E (K_{Alice}, R2)).
vii. Bob computes $K_{Bob} = (R_1)^{y}$ mod p, and calculate (C2 = E (K_{Bob}, R_2)).
viii. Bob (the verifier) verify ($C_1 = C_2$); if equal then Alice is accepted, otherwise it is rejected.

We discussed in section III security of D-H Key of Exchange Algorithm. There are two problems are encountered first one is discrete logarithmic attack and second one is man in the middle attack. To overcome these difficulties in existed algorithm, we are proposing two newer versions. Accordingly, we develop a first version based on the basic D-H key exchange algorithm which is vulnerable to man-in-the-middle-attack.Figure.4 man in the middle attack solution in D-H Key Exchange Algorithm.

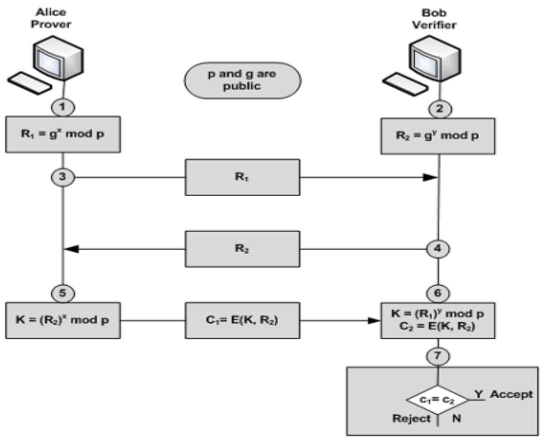

Fig.4 Proposed ZKP Version1

- ***Mathematical Analysis of Proposed ZKP Version-1:***

Let us consider
Alice private number = x = 6
Bob private number = y = 8
Public Number = p = 5
Public Number = g = 17
Calculating R_1 & R_2
$R_1 = g^x$ mod p.
⇨ $(17)^6$ mod 5
⇨ 24137569 mod 5
⇨ 4
Therefore **$R_1 = 4$**
$R_2 = g^y$ mod p.
⇨ $(17)^8$ mod 5
⇨ 6975757441 mod 5
⇨ 1
Therefore **$R_2 = 1$**
Calculating K_{Alice}, K_{Bob}
$K_{Alice} = (R_1)^x$ mod p
⇨ $(1)^6$ mod 5
⇨ 1 mod 5
Therefore **$K_{Alice} = 1$**
$K_{Bob} = (R_1)^y$ mod p
⇨ $(4)^6$ mod 5
⇨ 4096 mod 5
Therefore **$K_{Bob} = 1$**
Calculating C_1& C_2
C_1=E (K_{Alice}, R2)
We know
K_{Alice} = 1 = 0001
R_2 = 1 = 0001
Therefore $C_1 = K_{Alice}$, **XOR** R2
⇨ 0001 **Ex OR** 0001
C_1 = 0000
C_2=E (K_{Bob}, R_1)
We know
K_{Bob} = 1 = 0001
R_1= 1 = 0001
Therefore $C_2 = K_{Bob}$, **XOR** R_1
⇨ 0001 **Ex OR** 0001
C_2 = 0000

Hence as per the algorithm of proposed ZKP version 1, cipher texts of both parties are equal, then only key will accepts otherwise it will rejects. Above mathematical analysis will prove the same. In section V proposed version-1 algorithms will be verified using Xilinx 9.2i simulator and simulation results will be verified accordingly

[B] Proposed ZKP Version-2:

To protect the proposed algorithm from the man-in-the middle attack an encrypted replies (R1 and R2), and mutual authentication between the prover (Alice) and the verifier (Bob) is required.`

i. Alice (the prover) chooses a large random number x, such that 0< x < p and calculate R1 = g^x mod p.
ii. Alice sends R1 to Bob.
iii. Bob (the verifier) chooses another large random number y, such that 0< y < p and calculate R2 = g^y mod p, K_{Bob} = $(R1)^y$ mod p, and C1 = E (K_{Bob}, R2).
iv. Bob sends (R2 | C1) to Alice.
v. Alice, calculates K_{Alice} = $(R2)^x$ mod p, decrypt (R2' = D(K_{Alice}, C1)) and verify (R2 = R2') . If they matched then she proceeds; otherwise the verifier is dishonest.
vi. Alice encrypt (C2 = E (K_{Alice}, R1|R2) and send it to Bob.
vii. Bob decrypt C2 to get R1' and R2'
viii. Bob verify (R1 = R1'); if they are equal then Alice is verified (Accepted), otherwise it is a dishonest prover (rejected).

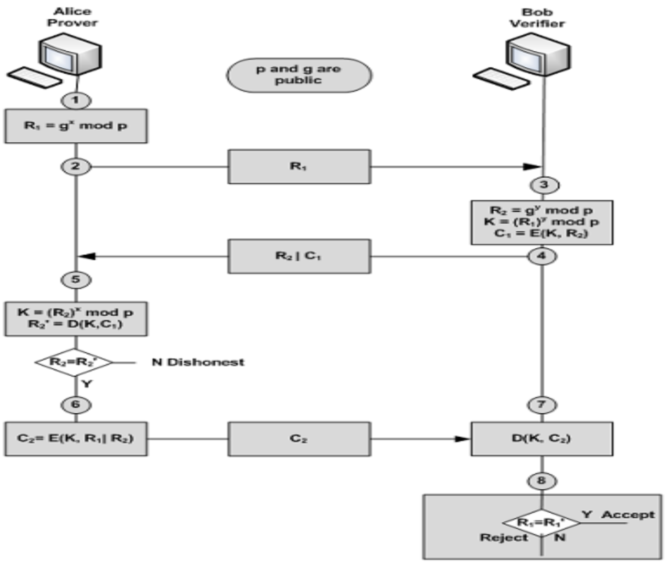

Fig.5 Proposed ZKP Versions 2

The prover (Alice) proves to the verifier (Bob) that she knows a secret by calculating the key (K) and resend Bob's reply (R2) to the verifier (Bob) encrypted with the generated secret key (K). Bob will encrypt his own reply (R2) with the generated secret key (K) and match the two encrypted information, if matched then Alice is verified, otherwise it is rejected. The verifier also needs to prove to the prover that he is honest by sending his reply R1togather with encrypted R1, then the verifier decrypt R1' by his key and match R1and R1', if they matched then the verifier is honest.

- ***Mathematical Analysis Proposed ZKP Version-2:***

Let us consider,
Alice private number = x = 6
Bob private number = y = 8
Public Number = p = 5
Public Number = g = 17

Calculating R_1 & R_2
$R_1 = g^x$ mod p.
⇨ $(17)^6$ mod 5
⇨ 24137569 mod 5
⇨ 4
Therefore **R_1 = 4**

$R_2 = g^y$ mod p.
⇨ $(17)^8$ mod 5
⇨ 6975757441 mod 5
⇨ 1
Therefore **$R_2 = 1$**
Calculating K_{Alice}, K_{Bob}
$K_{Alice} = (R_1)^x$ mod p
⇨ $(1)^6$ mod 5
⇨ 1 mod 5
Therefore **$K_{Alice} = 1$**
$K_{Bob} = (R_1)^y$ mod p
⇨ $(4)^6$ mod 5
⇨ 4096 mod 5
Therefore **$K_{Bob} = 1$**

Calculating C_1& C_2
C_1=E (K_{Alice}, R2)
We know
$K_{Alice} = 1 = 0001$
$R_2 = 1 = 0001$
Therefore $C_1 = K_{Alice}$, **Ex OR** R2
⇨ 0001 **Ex OR** 0001
$C_1 = 0000$

C_2=E (K_{Bob}, R_1)
We know
$K_{Bob} = 1 = 0001$
$R_1 = 1 = 0001$
Therefore $C_2 = K_{Bob}$, **XOR** R_1
⇨ 0001 X**OR** ~~0001~~ R1 = 4
c2 = 5 ~~**$C_2 = 0000$**~~

Calculating $R_1' = R_2'$

R_1' = D (K_{Bob}, C_2)
We know
$K_{Bob} = 1 = 0001$
$C_2 = 0 = 0000$
⇨ R_1' = D (K_{Bob}, C_2)
⇨ 0001 **Ex OR** 0000
Therefore **$R_1' = 1$**
R_2' = D (K_{Ailce}, C_1)
We know
$K_{Alice} = 1 = 0001$
$C_1 = 0 = 0000$
⇨ R_2' = D (K_{Ailce}, C_1)
⇨ 0001 **Ex OR** 0000
Therefore **$R_2' = 1$**

V. Fpga Implementation Of Proposed Work

[A] Specifications:

FPGA used in our proposed work is performance compared based on synthesis and simulation results to understand the device utilization and timing simulation by targeting on Xilinx Spartan 3S500E. For simulation and synthesis Xilinx ISE tool is used. Input the net list file generated from synthesizing, placing and routing on the Xilinx ISE 9.2i software. The simulation waveform for modification of DH Key Exchange Algorithm of proposed versions 1 and 2 are under the simulation clock 217.533 MHz's The waveform simulation takes place with 20 ns clock period.

[B] Simulation Results–Proposed Version1, 2

Fig.6 shows top module of proposed versions 1,2, which specifies a system with certain input’s and output. The inputs are given as per the specifications where clock can be generated as per internal clock generator automatically. Fig.7 describes the simulation waveform of proposed version 1,2.

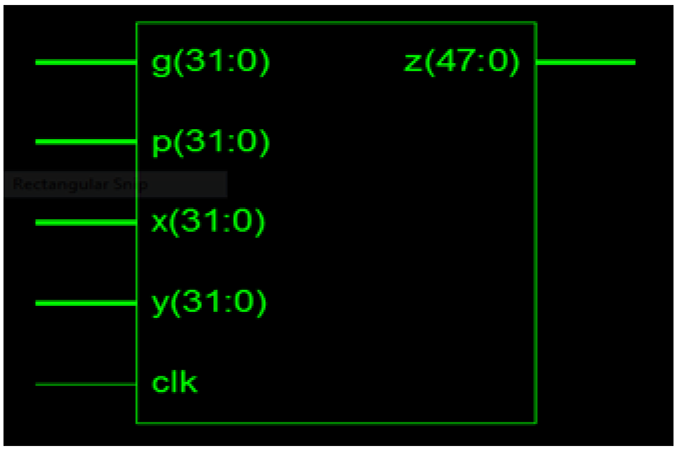

Fig. 6 Top Module of Proposed Versions1,2.

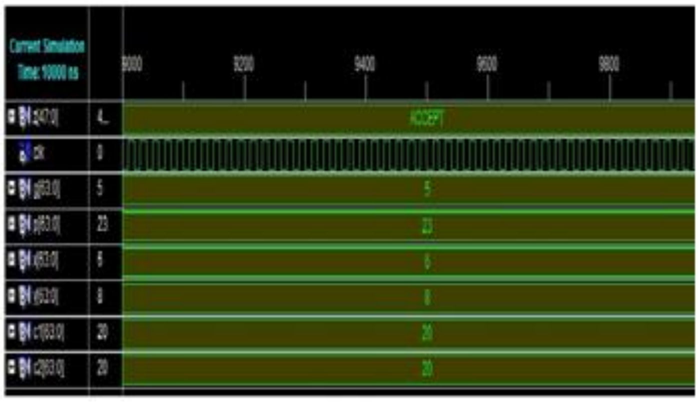

Fig. 7 Simulation Wave forms

Perhaps while doing simulation of proposed version 1 and 2 the results will show ACCEPT or REJECT as per the protocol existence. But the result is in ASCII format. In ASCII, each character consist of 8 bits, but the words ACCEPT or REJECT which has 6 characters each, so it needs total 48 bits to show output on target board. But our target board doesn’t support to show 48 bits at a time. That is the reason we took characters individually to show the version 1, 2 outputs as shown in Fig.8

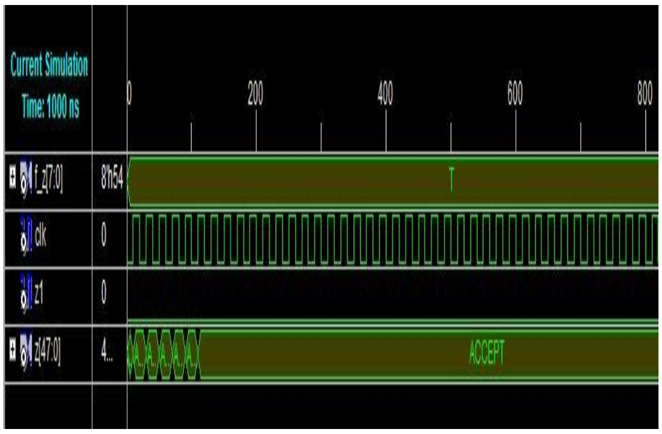

Fig.8 Simulation Waveform of Proposed version 1, 2 for FPGA implementation

[C] RTL and Technological Schematics:

Fig. 9 shows a Hierarchical Boundaries which explains internal schematic of an encoder within terms of register level. It also explains the transfer levels of inputs specified in the program and generating of registers. Fig.10 explains technology schematic of Proposed Versions 1 and 2 where it is flexible in showing the LUT's with reference to the FSM's. Lookup tables can be implemented with a multiplexer whose select lines are the inputs of the LUT and whose inputs are constants with respective of the specifications.

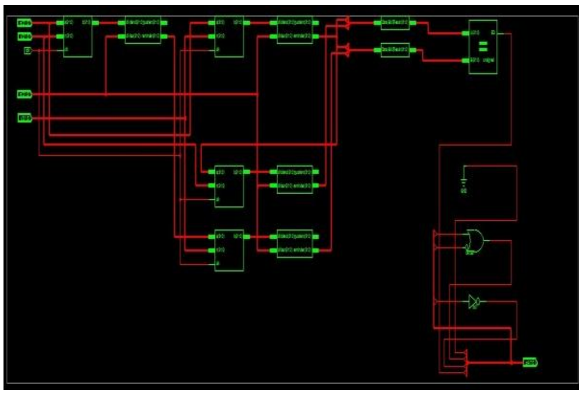

Fig.9 RTL Schematic

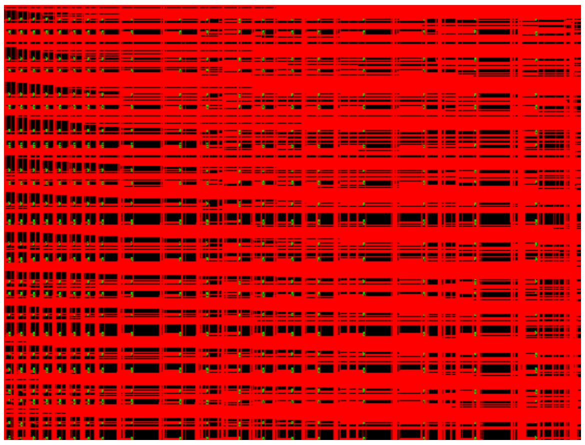

Fig.10 Technology Schematic

[D] Device Utilization and Timing Summary

Device Utilization Summary
Selected Device: 3s500efg320-4
Number of Slices: 6562 out of 4656 140% *
Number of Slice Flip Flops: 256 out of 9312 2%
Number of 4 input LUTs: 12656 out of 9312 135% (*)
Number of IOs:177
Number of bonded IOBs: 177 out of 232 76%
Number of MULT18X18SIOs: 12 out of 20 60%
Number of GCLKs: 1 out of 24 4%
Timing Summary:
Minimum period: 223.577ns (Maximum Frequency: 4.473MHz)
Minimum input arrival time before clock: 224.751ns
Maximum output required time after clock: 194.442ns
Maximum combinational path delay: 195.461ns

[D] Emulation Results-Proposed Version 1, 2

Fig.11 shows specified code for Proposed Version 1 or 2 to emulate into the kit with input specifications of "01010100", which is equivalent to letter "T" in ASCII. However the output shown here gives a serial output with LED glowing (green light at right bottom).

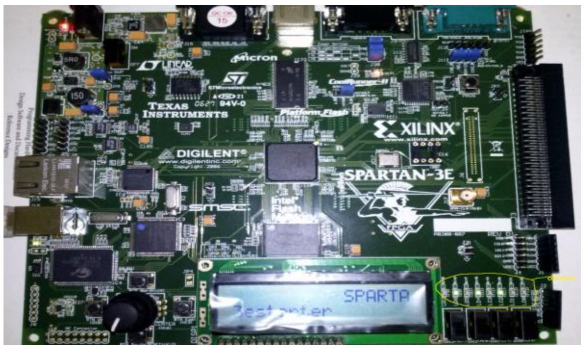

Fig. 11 Xilinx Spartan 3E Implementation

VI. Conclusion and Future Scope

Zero-knowledge proofs are probabilistic proofs because there is some small probability (soundness error) that allows a cheating prover to convince the verifier of a false statement. Standard techniques used to decrease the soundness error to any arbitrarily small value, but with additional computation cost. The proposed protocol is a deterministic algorithm with bounded values (not probabilistic), hence has no soundness error and no additional computation cost. The proposed protocol fulfills the ZKP properties and protected against discrete logarithm attack and man-in-the middle attack. The proposed algorithm serves as key exchange algorithm with the addition to authentication services.

Elliptic curve Diffie-Hellman (ECDH) is a relatively new key agreement algorithm based on Diffie-Hellman but using the elliptic-curve cryptography. Elliptic key operates on smaller key size. A 160-bit key in ECC is considered to be as secured as a 1024 bit key in Diffie-Hellman. For generating a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters - certain public constants that are shared between parties involved in secured and trusted ECC communication. This includes curve parameter a, b, a generator point G in the chosen curve, the modulus p, order of the curve n and the cofactor h. There are several standard domain parameters defined by SEC, Standards for Efficient Cryptography.

References

[1]. Ibrahem, M.K. (2012), “Modification of Diffie Hellman Key Exchange Algorithm**”** Future Communication Networks (ICFCN), 2012 International Conference on 2-5 April 2012, p.p. 147 -152.
[2]. Emmanuel Bresson, Olivier Chevassut, David Pointcheval, Jean-Jacques Quisquater “Provably Authenticated Group Diffie-Hellman Key Exchange” Computer and Communication Security, Proc. of ACM CCS’01, p.p. 255-264.
[3]. Back, Amanda, (2009), "The Diffe-Hellman Key Exchange", December 2, 2009, http://129.81.170.14/~erowland/courses/2009-2/projects/Back.pdf.
[4]. Clausen, Andrew, (2007), "Logical Composition of Zero-Knowledge Proofs", October, 2007, http://www.econ.upenn.edu/_clausen.
[5]. 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, October 2009.
[6]. Fischer, Michael J., (2010), "Cryptography and Computer Security", Department of Computer Science, Yale University, March 29, 2010.
[7]. Forouzan, Behrouz A. (2008), "Cryptography and Network Security", McGraw-Hill, Int. Ed. 2008.
[8]. Hellman, Martin E., (2002), "An Overview of Public Key Cryptography", IEEE Communications Magazine, May 2002, pp: 42-49.
[9]. Kizza, Joseph M, (2010), "Feige-Fiat-Shamir ZKP Scheme Revisited", International Journal of Computing and ICT Research, Vol. 4, No. 1, June 2010.
[10]. Maurer Ueli, (2009), "Unifying Zero-Knowledge Proofs of Knowledge", Africacrypt 2009, LNCS 5580, 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 (2007), "A Survey of Zero-Knowledge Proofs with Applications to Cryptography", Southern Illinois University, 2007.
[13]. P. Bhattacharya, M. Debbabi and H. Otrok, (2005), "Improving the Diffie-Heliman Secure Key Exchange", International Conference on Wireless Networks, Communications and Mobile Computing, 2005.
[14]. Simari, Gerardo I., (2002), "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, (2006), "Zero-Knowledge The Magic of Cryptography", Saarland University, August, 2006.
[17]. Krantz, Steven G., (2007), "Zero Knowledge Proofs", AIM Preprint Series, Volume 10-46, July25, 2007.
[18]. Carts, David A., (2001), "A Review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols", SANS Institute, 2001.

D.Surendra Rao. "FPGA Implementation of Modified Diffie-Hellman Key Exchange Algorithm Using Zero Knowledge Proof" IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) , vol. 9, no. 6, 2019, pp. 15-24.