Modification of Diffie–Hellman Key Exchange Algorithm for Zero Knowledge Proof

Dr. Mahmood Khalel Ibrahim
College of Information Engineering, University of Al Nahrain/ Baghdad
Email :mahmood_khalel@.com

Received on: 8/9/2011 & Accepted on: 5/1/2012

ABSTRACT

Networks and entity groupings requires entity authentication while preserving the privacy of the entity being authenticated. 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. This paper discusses zero-knowledge protocols and Diffie–Hellman algorithm and analyzes their vulnerability against known attacks. Also it presents a proposed protocol based on modification of Diffie–Hellman algorithm into an interactive zero-knowledge proof protocol. The proposed protocol is designed and developed to satisfy the zero-knowledge proof properties and resists the known attacks.

تعديل خوارزمية دفي-هلمن لتبادل المفاتيح الى خوارزمية المعرفه الصفرية

الخلاصة

تحتاج الشبكات ومجموعات المشتركين في هذه الشبكات الى خدمة التخويل مع المحافظة على خصوصية كل مستخدم بدون كشف معلومات سرية. و تلعب خوارزميات المعرفه الصفرية دوراً مهماً في توفير خدمة التخويل بدون كشف المعلومات السرية للمشتركين. كما توفر خوارزمية دفي-هلمن خدمة تبادل المفاتيح السرية في القنوات الاعتيادية غير المحمية. في هذا البحث تم مناقشة خوارزمية المعرفه الصفرية وخوارزمية دفي-هلمن وتحليل نقاط الوهن فيها تجاه الهجمات المعروفة. كما يقدم البحث خوارزمية مقترحة لتعديل خوارزمية دفي-هلمن الى خوارزمية المعرفه الصفرية. تم تصميم هذه الخوارزمية لتحقق الخواص المطلوبة في المعرفه الصفرية مع خاصية مقاومة الهجمات المعروفة.

INTRODUCTION

In simple password protocols, a claimant *A* gives his password to a verifier *B*. If certain precautions are not taken, an eavesdropper can get the password that was transferred, and from there on he can impersonate *A* to his benefit. Other protocols try to improve on this, as in the case of challenge-response systems. In this sort of protocols, *A* responds to *B*'s challenge to prove knowledge of a shared secret [14].

https://doi.org/10.30684/etj.30.3.9
2412-0758/University of Technology-Iraq, Baghdad, Iraq

A zero-knowledge proof is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the verity of the statement. It is common practice to label the two parties in a zero-knowledge proof as the *prover* of the statement and the *verifier* of the statement. Sometimes *P* and *V* are known instead [16]. A common use of a zero-knowledge proof is in authentication systems where an entity proves his identity to the prover without revealing his secret [12].

Diffie–Hellman (D-H) key exchange algorithm is a specific method of exchanging secret keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The D–H key exchange algorithm allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communication channels. This key can then be used to encrypt subsequent communications using a symmetric key cipher. It is designed specially to exchange secret key in insecure communication channels [2, 5].

In this paper zero-knowledge proof, Fiat-Shamir ZKP Protocol and D–H key exchange algorithm has been presented and criticized. A new ZKP has been proposed based on modification of the D–H key exchange algorithm to a zero-knowledge protocol. Two versions of the proposed protocol are presented; the first one was built around the basic D-H key exchange algorithm, which is vulnerable to man-in-the-middle-attack. The second proposed version solves the problem of the mentioned attack.

ZERO-KNOWLEDGE PROOF

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 [4].

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 [12].

One of the most fascinating uses of zero-knowledge proofs within cryptographic protocols is to enforce honest behavior while maintaining privacy. Roughly, the idea is to enforce a user to prove, using a zero-knowledge proof, that its behavior 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 [10, 9].

Definition of Zero Knowledge Proof

ZKP model of computation defined as an interactive proof system (P,V), where P is a *prover* and V is a *verifier*. Protocol (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 $x \in L$, 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 in $|x|$ and proof instance $(P,V)(x)$ must terminate in polynomial time in $|x|$. Upon completing the interaction, the output of the protocol should be of form $(P,V)(x) \in \{Accept, Reject\}$ representing *V's* acceptance or rejection of *P's* claim that $x \in L$ [5].

Three properties are expected from a zero-knowledge proof [11, 3]:

a. Completeness: An interactive proof (protocol) is complete if, given an honest *prover* and an honest *verifier* (that is, one following the protocol properly), the protocol succeeds with overwhelming probability (i.e., the verifier accepts the prover's claim).

b. Soundness: An interactive proof (protocol) is sound if there exists an expected polynomial time algorithm M with the following property: if a dishonest *prover* (impersonating P) can with non-negligible probability successfully execute the protocol with V, then M can be used to extract from this prover knowledge (essentially equivalent to P's secret) which with overwhelming probability allows successful subsequent protocol executions.

c. Zero-knowledge: a protocol has zero-knowledge property if it is simulatable in the following sense; there exists an expected polynomial-time algorithm (*simulator*) which can produce, upon input of the assertion(s) to be proven but without interacting with the real prover, transcripts indistinguishable from those resulting from interaction with the real prover.

Zero-knowledge proofs are not proofs in the mathematical sense of the term, because there is some small probability (called the *soundness error*) that a cheating *prover* will be able to convince the *verifier* of a false statement. However, there are standard techniques to decrease the soundness error to any arbitrarily small value [4, 6].

Fiat-Shamir ZKP Protocol

In cryptography, theFiat-Shamir identification scheme is a type of interactive zero-knowledge proof. Like all zero-knowledge proofs, the Fiat-Shamir scheme allows one party (*prover*), to prove to another party (*verifier*), that he possesses secret information without revealing to him what that secret information is [8].

In Fiat-Shamir protocol, a trusted third party selects two large prime numbers p and q

to calculate the value of ($n = p.q$). The value of n is announced to the public; the values p and q are kept secret. *Alice* the *prover* choose a secret number ($1< s< n-1$) and calculate ($v = s^2 \ mod \ n$). She keeps s as private key and register v as her public key with the third party. Figure-1 illustrates the steps of the protocol. *Alice*, the *prover* and *Bob* the *verifier* performs the following procedure [6]:

1. *Alice*, the *prover*, chooses a random number r (commitment) such that ($1£ \ r£ \ n-1$), she then calculate the value of ($x = r^2 \ mod \ n$), x called the witness.
2. *Alice* sends x to *Bob* as the witness.
3. *Bob*, the *verifier*, sends the challenge c to *Alice*. The value of c is [$0, 1$].
4. *Alice* calculates the response ($y = rs^c$), where s is *Alice's* private key.
5. *Alice* sends the response (y) to *Bob* to prove that she knows her private key (she claims to be *Alice*).
6. *Bob* calculate y^2 and xv^c. If these two values are congruent, then *Alice* either knows the value of s (honest) or she calculated y in some other way (dishonest).

$$[y^2 \ mod \ n = (rs^c)^2 \ mod \ n = r^2s^{2c} \ mod \ n = r^2(s^2)^c \ mod \ n = xv^c \ mod \ n]$$

Repeat steps (1- 6) several times with value of c equal to *0 or 1*. The *prover* must pass the test in each round to be verified.

Feige-Fiat-Shamir protocol is similar to Fiat-Shamir protocol except that it uses a vector of private keys [s_1, s_2, … s_k], a vector of public keys [v_1, v_2, …, v_k], and a vector of challenges [c_1, c_2, …, c_k]. The keys are chosen randomly but they must be relatively prime to *n*. TheFeige-Fiat-Shamir Identification Scheme, however, uses modular arithmetic and a parallel verification process that limits the number of communications between the prover and verifier **[6, 8].**

Zero Knowledge Proof Analysis

Zero-Knowledge protocols can be fooled by a third party user (*Eve*) pretending she is actually *Alice* to *Bob* and therefore get access. *Eve* just has to guess the challenge (as mentioned above, c can only be 0 or 1) and send the response y which is set to a random number without using the secret key for encryption $y = r$). Two situations can happen [6, 8]:

a. *Eve* guesses that the value ($c = 1$), *Eve* calculates ($x = r^2/v$) and sends x as witness. If her guess is correct then she sends ($y = r$) as the response and pass the test because; ($y^2 = r^2$and $y^2 = xv^c = r^2v^c/v = r^2v^1/v = r^2$).
b. *Eve* guesses that the value ($c = 0$), *Eve* calculates ($x = r^2$) and sends x as witness. If her guess is correct then she sends ($y = r$) as the response and pass the test because; ($y^2 = r^2$and $y^2 = xv^c = r^2v^0 = r^2$).

This works perfectly for the Fiat-Shamir-Scheme where the chance of guessing correctly is about 50:50. However, if the process is repeated 20 times, the probability of *Eve* correctly guessing *Alice's* secret number decreases to $(1/2)^{20} \cong 9.54\text{x}10^{-7}$ [8].Of course, the challenge is changed every time the protocol is used; therefore, an Eavesdropper can, in time, gather enough partial information about the shared Secret to try an impersonation attack like the one described above [8].

Recall the definition of ZKP, then *(P,V)(x)* is a probabilistic system such that; For each *x*, output value *(P,V)(x)* is a random variable of common input *x*, private input value of *P*, and some random input values of *P* and *V* [5,6].

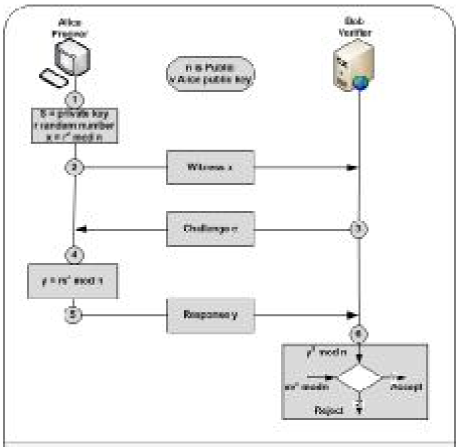

Figure-1 Fiat-Shamir Protocol

DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

Diffie-Hellman key exchange algorithm was invented in 1976 during 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 $p<Z_{p*},x>$, 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 $<Z_{p*},x>$ 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 [6, 7, 13]:

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

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

$$\boldsymbol{K_{Alice} = (R_2)^x modp = (g^y \; mod \; p)^x mod \; p= g^{xy} \; mod \; p.}$$

$$K_{Bob} = (R_1)^y modp = (g^x \; mod \; p)^y mod \; p= g^{xy} \; mod \; p.$$

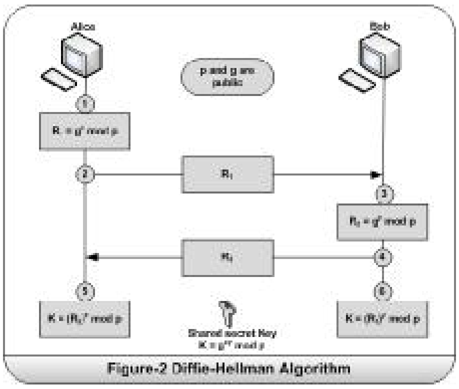

Figure-2 Diffie-Hellman Algorithm

SECURITY OF DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM

The Diffie-Hellman algorithm is susceptible to two attacks; the discrete logarithm attack and the man-in-the-middle attack [6].

Discrete Logarithm Attack

An interceptor (*Eve*) can intercept R_1 and R_2 and [6, 15];

Find x from ($R_1 = g^x \; mod \; p$);

Find y from ($R_2 = g^y \; mod \; p$);

Then she can calculate ($K = g^{xy} \; mod \; p$). The secret key is not secret anymore.

To make Diffie-Hellmansafe from the discrete logarithm attack, the following are recommended:

a. The prime number p must be very large (more than 300 digits).

b. The generator g must be chosen from the group $<Z_{p*},x>$.

c. The numbers x and y 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 [15, 1].

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 shows the man-in-the-middle attack. The attack can be performed as follows [6, 15]:

1. *Alice* chooses *x*, and calculate $R_1 = g^x \ mod \ p$ and sends $R_1$to *Bob*.
2. *Eve*, the intruder, intercept R_1, chooses *z*, calculates $R_2 = g^z \ mod \ p$, send R_2 to both *Alice* and *Bob*.
3. *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*.
4. *Alice* and *Eve* calculate $K_1= g^{xz} \ mod \ p$, which becomes shared key between them.
5. *Eve* and *Bob* calculate $K_2 = g^{zy} \ mod \ p$, which becomes shared key between them.
6. However, man-in-the-middle attack can be prevented by a station-to-station key agreement by using digital signature with public key certificates to establish a session key between *Alice* and *Bob* [6, 1].

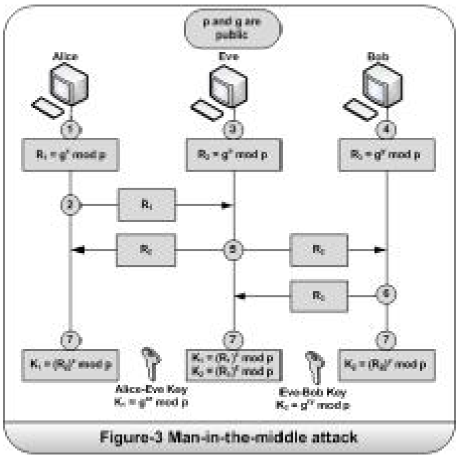

Figure-3 Man-in-the-middle attack

PROPOSED ZKP

The proposed ZKP based on D-H key exchangealgorithm 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 exchangealgorithm 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 in the next two articles.

Proposed ZKP Version-1

A trusted third party selects two prime numbers *p* and *g*, and announced as public numbers. Where *p* (modulo) is a large prime number and *g* is a primitive root of order (*p*-*1*) in the group $<Z_{p*},x>$.

The *prover* (*Alice*) proves to the *verifier* (*Bob*) that she knows a secret by calculating the key (K_{Alice}) and resend *Bob's* reply (R_2) to the *verifier* (*Bob*) encrypted with the generated secret key (K_{Alice}). *Bob* will encrypt his own reply (R_2) with the generated secret key (K_{Bob}) and match the two encrypted information; if they matched then *Alice* is verified, otherwise it is rejected. Figure-4 shows the procedure of the proposed protocol. The protocol performed as follows:

1. *Alice* (the *prover*) chooses a large random number *x*, such that $0< x < p$ and calculate $R_1 = g^x \mod p$.
2. *Bob* (the *verifier*) chooses another large random number *y*, such that $0< y < p$ and calculate $R_2 = g^y \mod p$.
3. *Alice* sends R_1 to *Bob*.
4. *Bob* sends R_2 to *Alice*.
5. *Alice* (the *prover*), computes $K_{Alice}= (R_2)^x \mod p$, and send encrypted R_2 to *Bob* using $K_{Alice}(C_1 = E(K_{Alice}, R_2))$.
6. *Bob* computes $K_{Bob} = (R_1)^y \mod p$, and calculate ($C_2 = E(K_{Bob}, R_2)$).

Bob (the *verifier*) verify ($C_1 = C_2$); if equal then *Alice* is *accepted*, otherwise it is *rejected*.

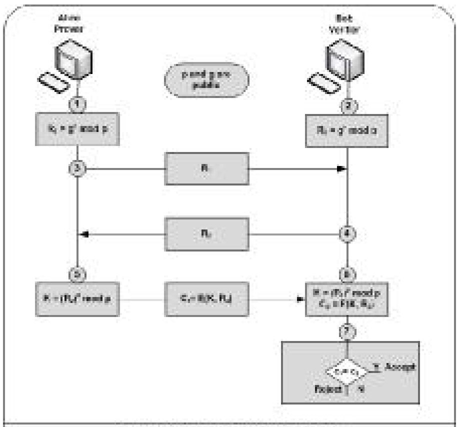

Figure-4 Proposed ZKP Version-1

Proposed ZKP Version-2

The proposed algorithm version-1 is vulnerable against man-in-the-middle attack. An eavesdropper *Eve* can do the following:

Eve select a random number *z*, and intercept R_1 of *Alice* and R_2 of *Bob*. *Eve* calculate two secret keys; ($K_1 = R_1^z \mod p$) to be shared with *Alice*, and ($K_2 = R_2^z \mod p$) to be shared with *Bob*. Eve sends ($C_1 = E(K_1, R_1) \mod p$) to *Alice* and sends ($C_2 = E(K_2, R_2) \mod p$) to *Bob*. Hence *Eve* can impersonate *Alice* and verified with *Bob*.

To protect the algorithm from the man-in-the-middle attack an encrypted replies (*R1* and R_2), and mutual authentication between the *prover* (*Alice*) and the *verifier* (*Bob*) is required.

The *prover* (*Alice*) proves to the *verifier* (*Bob*) that she knows a secret by calculating the key (*K*) and resend *Bob's* reply (R_2) to the *verifier* (*Bob*) encrypted with the generated secret key (*K*). *Bob* will encrypt his own reply (R_2) with the generated secret key (*K*) and match the two encrypted information, if matched then *Alice* is verified, otherwise it is rejected. Figure-5 shows the procedure of the proposed protocol.

The security of the proposed algorithm based on the idea that after intercepting *R1*, *Eve* cannot send her own *R2* to *Alice* and pretend it is coming from *Bob*, because *Eve* cannot forge the key of *Bob* to create *C1*. In the same way, *Eve* cannot forge *Alice* key to create *C2*. The protocol performed as follows:

1. *Alice* (the *prover*) chooses a large random number *x*, such that $0< x < p$ and calculate $R_1 = g^x \mod p$.
2. *Alice* sends R_1 to *Bob*.
3. *Bob* (the *verifier*) chooses another large random number *y*, such that $0< y < p$ and calculate $R_2 = g^y \mod p$, $K_{Bob} = (R_1)^y modp$, and $C_1 = E(K_{Bob}, R_2)$.
4. *Bob* sends ($R_2 \mid C_1$) to *Alice*.
5. *Alice*, calculates $K_{Alice}= (R_2)^x modp$, decrypt ($R_2' = D(K_{Alice}, C_1)$) and verify($R2 = R2'$). If they matched then the 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*).

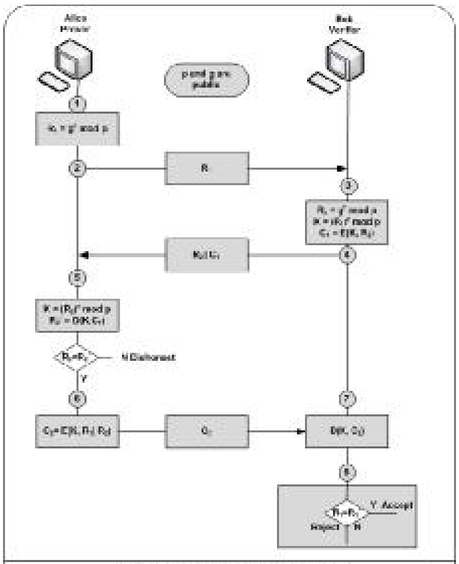

Figure-5 Proposed ZKP Version-2

Analysis of the Proposed Protocol

Recall the definition of ZKP discussed in (2.1), the proposed protocol is an interactive proof system (*P*, *V*) for proving a language membership statement for a language over {*G*}, where *G* is a group $<Z_{p*},x>$. R_1, R_2, C_1, and C_2 are a membership

instances $\in G$ and proof transcript. *P* and *V* are exchanging a finite sequence of proof transcript and upon completion the interaction the output of the protocol will be {*Accept or reject*}. The proposed protocol satisfies the ZKP properties as follows:

a. Completeness: if *Alice* (the *prover*) and *Bob* (the *verifier*) are honest, then on performing the protocol steps, it must ends with {*accept*, *reject*} decision. That is because the final decision depends on the computed value of the secret key *K* which is equal to ($K = g^{xy} \bmod p$) on both sides, which can be either identical [*accept*] or different [*reject*].

b. Soundness: if the *prover* fail to compute the correct value of the secret key *K*, the encrypted reply ($C_2' = E(K_{Alice}, R_2)$) will be different from the value computed by the *verifier*($C_2 = E(K_{Bob}, R_2)$).

The values of C_1 and C_2 can't be guessed, there are two possibilities; either ($C_2 = C_2'$) or ($C_2 \neq C_2'$). The proposed algorithm is not a probabilistic protocol; hence it has no soundness error.

c. Zero-Knowledge: on completion, both parties; the *prover* and the *verifier* would not have any further information other than their own secret numbers and calculated secret key. Secret numbers *x*, *y* and *K* was not revealed.

d. The proposed protocol analysis follows the analysis of Diffie-Hellman Key Exchangealgorithm. It can be protected against discrete logarithm attack, by applying the recommendation mentioned in (4.1). The second version of the proposed protocol is protected from man-in-the-middle attack.

CONCLUSIONS

a. 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.

b. The proposed protocol is a deterministic algorithm with bounded values, hence has no soundness error and no additional computation cost.

c. The proposed protocol fulfills the ZKP properties and protected against discrete logarithm attack and man-in-the-middle attack.

d. Proposed algorithm serves as key exchange algorithm with the addition to authentication services.

REFERENCES

[1]Back, Amanda, (2009), "***The Diffie-Hellman Key Exchange***", 2009, http://129.81.170.14/~erowland/courses/2009-2/projects/Back.pdf, cited, January 2011.

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

[3]Clausen, Andrew, (2007), "***Logical Composition of Zero-Knowledge Proofs***", http://www.econ.upenn.edu/_clausen, cited, January 2011.

[4]EndreBangerter, 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 CryptographicCompilers (SPEED-CC'09), Berlin, Germany, October2009.

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

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

[7]Hellman, Martin E., (2002), "***An Overview of Public Key Cryptography***, IEEE Communications Magazine, May 2002, pp: 42-49**.**
http://austinmohr.com/work/files/zkp.pdf, cited January, 2010.

[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., (2007), **"*Zero Knowledge Proofs*",** AIM Preprint Series, Volume 10-46, July25, 2007.

[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***". http://austinmohr.com/work/files/zkp.pdf, cited January, 2010.

[13]P. Bhattacharya, M. Debbabi and H. Otrok, (2005), "***Improving the Diffie-Hellman 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**.**