Secure Messaging System using ZKP

Mahmood Khalel Ibrahem, Tamara Alaa M. Ali

Networks Engineering Department, College of Information Engineering, Al-Nahrain University, Baghdad, Iraq

Abstract- **In this paper, a web-based messaging system is presented with implementing a new version of Diffie-Hellman protocol, to insure mutual authentication between client and server along with exchanging key securely without sending it through the channel.**

The proposed messaging system provides the major security services which are Authenticity, Integrity and Confidentiality, by implementing Zero Knowledge Proof (ZKP) and password authentication for authenticity, Hmac for integrity and AES for Confidentiality.

***Keywords—Security services (Authenticity, Integrity and Confidentiality), Zero Knowledge Protocol, Diffie-Hellman, AES, Hmac* .**

I. INTRODUCTION

Websites are now the number one target of choice for attacks by hackers. Their attacks have moved from the well-defended network layer to the more accessible Web application layer that people use every day to manage their lives and transact business. The sites where consumers shop, bank, manage their healthcare, pay insurance, book travel and apply to college are now under a near-constant barrage of attacks intent upon stealing their credit card numbers and other personal private information [1].

II. WEB SECURITY CONSIDERATIONS

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. Web presents new challenges not generally appreciated in the context of computer and network security:

A. The Internet is two way. The Web is vulnerable to attacks on the Web servers over the Internet.

B. The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions.

C. Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws.

D. A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.

E. Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures [2].

III. ZERO KNOWLEDGE AUTHENTICATION

Zero knowledge authentication protocols are one of the most trusted authentication protocols. In zero knowledge authentications, the claimant does not reveal anything that might endanger the confidentiality of the secret. The claimant proves to the verifier that he/she knows a secret, without revealing it. The interactions are so designed that they cannot lead to revealing or guessing the secret. After exchanging messages, the verifier only know that the claimant does or does not have the secret, nothing more. The result is a yes/no situation, just a single bit of information [3].

"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.

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) [4].

P and V are linked by a communication channel over which they exchange a sequence, called proof transcript a1, b1, a2, b2... an, bn. Proof transcript interleaves prover's transcript and verifier's transcript. Each element ai, bi 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 [6]:

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 simulate-able 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.

IV. USED ZKP PROTOCOL

The Used ZKP based on Diffie-Hellman key exchange algorithm in the sense that both parties (the prover and the verifier) exchange non secret information without revealing secret information to get one identical secret key. This means that the prover can prove to the verifier that he knows the secret [7].

The verifier needs to prove to the prover that he is honest by sending his reply R_1 together with encrypted R_1, then the verifier decrypt $R_{1'}$ by his key and match R_1 and $R_{1'}$, if they matched then the verifier is honest. The prover (Alice) needs to prove 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.

The used algorithm has been developed to resists the man-in-the-middle attack. For more details refer to [7]. Figure 1 shows the procedure of the used ZKP protocol. The protocol performed as follows:

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

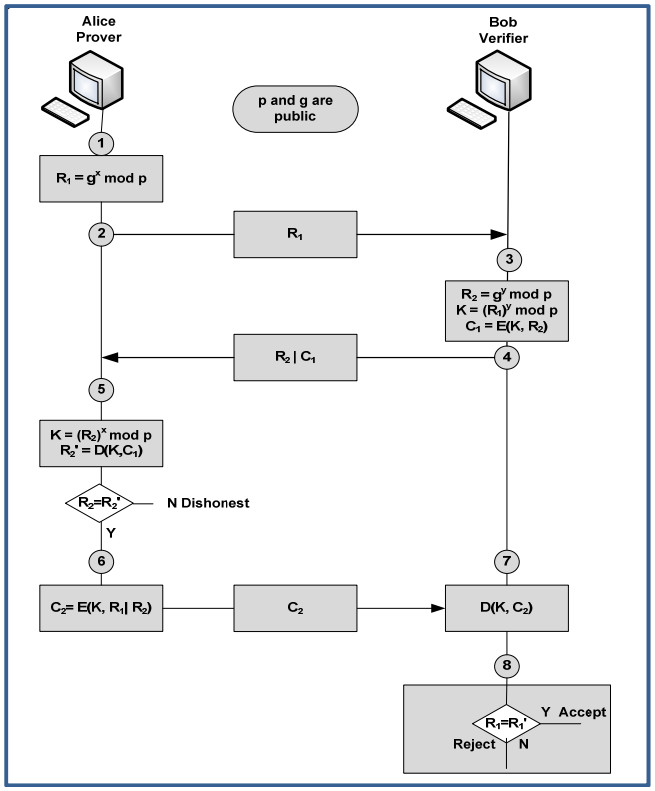

Fig. 1: Used ZKP protocol procedure

V. PROPOSED SYSTEM ARCHITECTURE

To achieve the system robustness, flexibility and resistance to potential changes, the popular three-tier (layer) architecture is deployed in the proposed system. The architecture is a web-based system composed of three layers: the user interface layer, the application logic layer and the database layer. The three-layer architecture aims to make the application development and implementation easier and more efficient. The interface layer in the three-layer architecture offers the user a friendly and convenient entry to communicate with the system while the application logic layer performs the controlling functionalities and manipulating the underlying logic connection of information flows; finally, the data modeling job is conducted by the database layer, which can store, index, manage and model information needed for this application [8].

Figure 2 shows a block diagram which illustrates the structure of the proposed system; in the next sections these components will be described in detail.

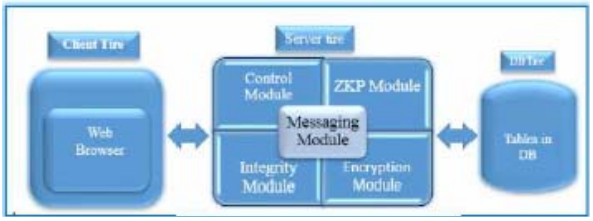

Fig. 2 General structure of the proposed system (3-tier architecture model)

The proposed system architecture is presented in figure 3 below, illustrate all the functions implemented in the architecture.

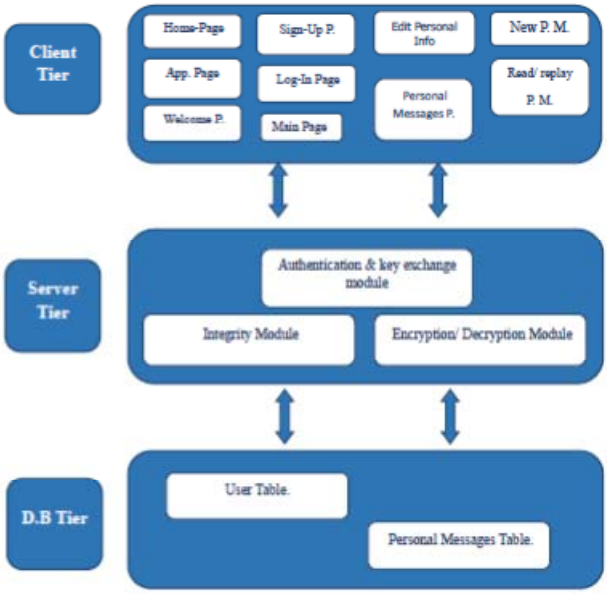

Fig. 3 Proposed system architecture

VI. CLIENT TIER

First step in the proposed security model is to load the *home web-page* (http://localhost/home.html), where the applicant should enter any random number *x* as a *secret number* which will be used in performing authentication and key exchange function.

Home page will pass the secret number to the authentication and key exchange module, which an implementation of the zero-knowledge proof that accomplish the authentication procedure with the addition to key exchange.

After a successful authentication of the user, the system will prompt a message informing the client for being authenticated, and displaying the available applications asking the client to select one of them, at the application web page Otherwise, if the client is not authenticated, an error message will appear to the user that illustrate he should try again. Application Page is responsible of controlling the access to the enterprise network and application.

At this step, the client is a legitimate client of the enterprise network. However, to access any application on the network server, the client will need additional authentication dedicated to each individual application.

Case study application (which is a *messaging system*) has been developed to demonstrate the access of the client to the enterprise application with addition to the security services.

The user name and password will get hashed and encrypted by means of HMAC and AES encryption algorithm respectively, beside that the user password will be scrambled by MD5.

Otherwise if the client does not have one, he should sign up in order to be able to get access at *Sing-Up page*.

Personal messages Pages, Messages related pages are*; create a new message page, list of messages page* and *reading/ replying messages page.*

This system provides to the user the ability of sending/receiving messages, in a secure way. Integrity and encryption modules need a key, which in this system is the key resulted from implementing ZKP authentication and key exchange protocol. These features will be achieved through the collaboration of the four modules which ZKP, messages, integrity, and encryption modules. Figure 4 shows the four modules collaboration.

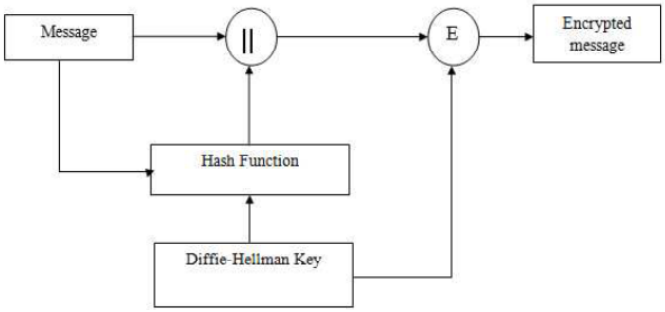

Fig. 4 Four modules collaboration

The figure above represents stages that each message should be through in order to be send to the precipitant, where each message will get through a hash function to get its Hmac, and then the resulted Hmac will be concatenated with the original message, the resulted data block will be encrypted using AES encryption algorithm. Both of hash function and AES encryption algorithm will use the key resulted from ZKP.

On the other side the receiver will do the reverse order of these steps with a kind of differences, where the receiver will do the decryption operation first on the receiver data, then from the resulted data Hmac function result will be extracted, and compute it for the received message. The system than will check if the receiver message is as same as it send by comparing Hmac values of the send one and the computed one, to accept it or not.

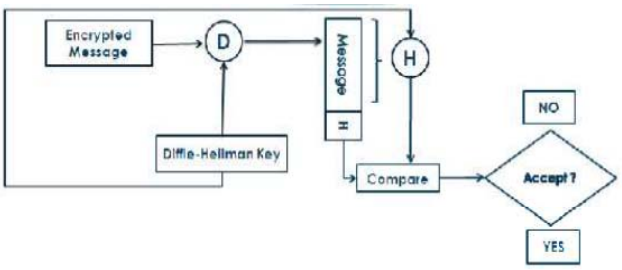

Fig. 5 Receiving a message steps

VII. SERVER TIER:

The server logic is moved out of the client code and placed in the middle tier (server tier). The basic concept of the three tier model is partitioning the system functionality in layers, so applications gain scalability and security. This tier presents the application logic layer, which bridges the gap between the user interfaces and the underlying database, hiding technical details from the users. Components in this layer receive requests coming from the interface layer and interpret the requests into apropos actions controlled by the defined work flow in accordance with certain pre-defined rules. Application logic layer consists of a controller module and four functional modules.

The modules are as described briefly in the following sections: Control module controls the flow of functions execution and transferring required information between them and the database tier, Figure 6 shows how the control module works.

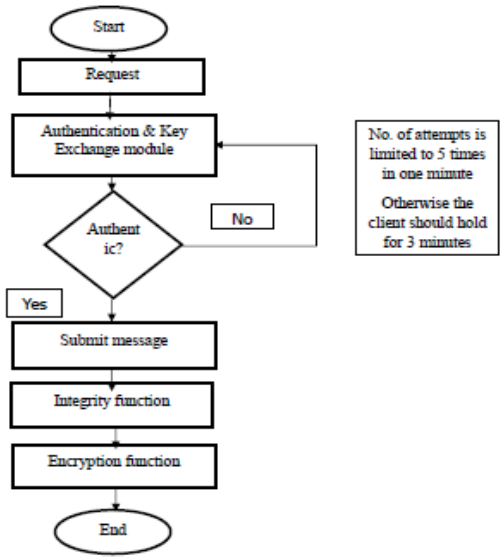

Fig. 6 Control module work chart

The functional modules are:

A. ZKP Authentication and Key Exchange Module

For illustration purpose figure 7 presents the Diffie-Hellman mathematical module. The client should enter his own secret number, which is *x*, at the same time the server will enter his own secret number which is *y*, the implemented protocol which is the modified Diffie-Hellman will calculate at each side R_1, R_2 at client and server side respectively, R_1 will passes the server side using the formula:

$$R_{1=}g^X \ mod \ p$$

R_2 will also pass to the client side along with C_1, in the following formula:

$$R_{2=}g^Y \ mod \ p$$
$$C_1 = E(K_1, R_2)$$

Next step is calculating K at each side using the following formula respectively at the server and client side:

$$K_1 = (R_1)^y mod \ p$$
$$K_2 = (R_2)^x mod \ p$$

The same value of K will be established in both sides. This value is the secret key, which will be used in implementing the integrity and confidentiality security services.

The interactions are designed in a way that cannot lead to revealing or guessing the secret of both of the interconnected parties. After exchanging values the prover will be either accepted, or rejected and halt the system. Exchanged messages between prover and verifier are encrypted by AES encryption algorithm using the generated key.

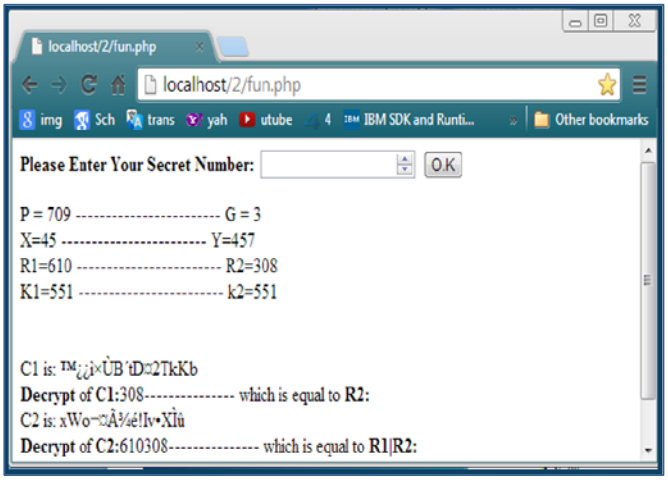

Fig. 7 Diffie-Hellman arithmetical module

B. Integrity Module

The third module is responsible of calculating messages hash, and attaching it to message after that, according to attached message hash, the client will make sure that these messages had not been altered anywhere in a not authorized way. This function is accomplished using hash function.

C. Encryption Module

The fourth module is the encryption module, which is responsible for:

I. Encrypting password by means of using MD5 function, in order to store it in its encrypted version to reduce security risks.

II. Encrypting user name and password in both sign-up, log-in process, and messages between the client and server in the messages system (case study application). It uses AES encryption algorithm along with the key resulted from the ZKP protocol.

For illustrative purpose figure 8 below demonstrates the implementation of encryption using AES encryption algorithm and integrity using HMAC in the Log-In process, where both of them represent the confidentiality and integrity services.

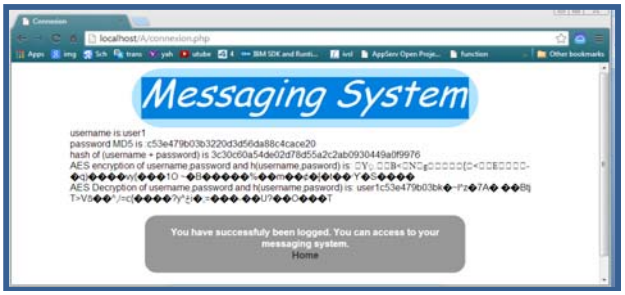

Fig. 8 Confidentiality and integrity implementation in the system example

In order to clarify the implementation of the confidentiality and integrity functions in messages related issues, Figure 4-16 below for illustrative purpose demonstrates the steps required to implement integrity and confidentiality respectively, assuming the message is "*TEST MESSAGE*", its HMAC will be "*96d572176b5a59bcc4ebff8d4640b6ee7fc05b77*", after concatenation the message with its HMAC, the resulted string will be encrypted through AES (advanced encryption standard) which will be:

"��nB��@�5S����)G�����g����. V����M�b+´ ̊e̊v̊q̊̊̊̊Y@̊̊̊Y".

Fig.9 Messages related integrity and confidentiality implementation example

VIII. Database Tier

This tier is responsible for modeling and storing information needed for the system and for optimizing the data access. Data needed by the messaging system logic layer will be retrieved from the database. The data-base of the proposed system consists of two tables in order to store information related to the client (Personal messages and account related information).

IX. 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 [9].

A. Discrete Logarithm Attack

An interceptor (Eve) can intercept R_1 and R_2 [2, 9];

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

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

Then she can calculate ($K = g^{xy} \bmod p$). The secret key is not secret anymore. To make Diffie-Hellman safe from the discrete logarithm attack, the following are recommended:

1) The prime number p must be very large (more than 300digits).
2) The generator g must be chosen from the group <Zp*, x>.
3) 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 [2, 10].

B. 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 10 shows the man-in-the middle attack. The attack can be performed as follows [2, 9]:

1) Alice chooses *x*, and calculate $R_1 = g^x \bmod p$ and sends R_1 to Bob.
2) Eve, the intruder, intercept R_1, chooses z, calculates $R_2 = g^z \bmod p$, send R_2 to both Alice and Bob.
3) Bob chooses *y*, and calculate $R_3 = g^y \bmod p$ and sends R_3 to Alice. R_3 is intercepted by Eve and never reaches Alice.
4) Alice and Eve calculate $K_1 = g^{xz} \bmod p$, which becomes shared key between them.
5) Eve and Bob calculate $K_2 = g^{zy} \bmod p$, which becomes shared key between them.

The proposed system is resists man-in-the-middle attack as long as it use the modified version of the Diffie_Hellman protocol, since Eve cannot calculate two secret keys; (K1 = R1 x mod p) and (K2 = R2 y mod p) to be shared with Alice and Bob.

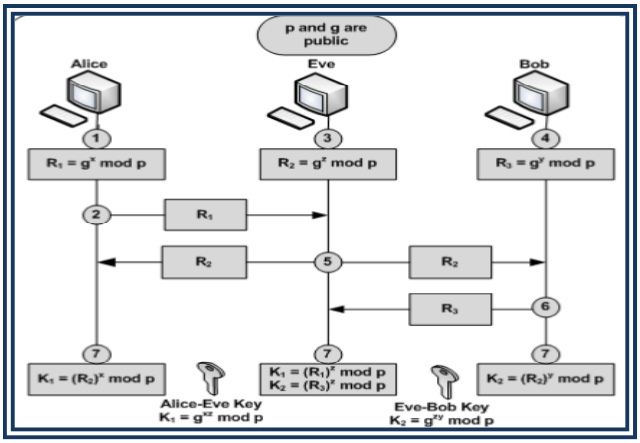

Fig. 10 Man in the middle attack

XII. CONCLUSIONS

In this paper, the security of an enterprise network was the major concern. The security issue has been discussed and considered in the design and implementation of the system into two categories;

The first category is the mutual authentication between the prover and the verifier which is an important issue as a security service requirement.

The second category is the integrity and confidentiality of the system which is another crucial issue to keep the data transmitted over unsecure channel unrevealed and unchanged.

The following points have been concluded throughout the work of the research:

1. The proposed protocol for Zero-knowledge proof is a deterministic algorithm with bounded values.
2. Implementing the proposed security model on Apache Web Server is much cheaper and easier than implementing it on Microsoft server where it's free.
3. On theoretical basis Man-In-the Middle attack and Discrete Logarithm attack for this system is not an issue any more.
4. It resists man-in-the-middle attack, since Eve cannot calculate two secret keys; (K1 = R1 x mod p) and (K2 = R2 y mod p) to be shared with Alice and Bob.
5. The major security goals (CIA) are satisfied and give confidence to users to communicate securely.

REFERENCES

[1] Jeremiah Grossman, "*10 Important Facts About Website Security and How They Impact Your Enterprise*", white paper, WhiteHat Security,January 2011.

[2] Stallings, William, "*Cryptography and Network Security*", Prentice Hall, 5th Ed. 2010.

[3] John E. Canavan, "*Fundamentals of Network Security*", first edition, Artech House, 2001.

[4] Dowd P.Winy; McHenry J.Time, "*Network security: it's time to take it seriously*"**,** IEEE Computer Security Society 31(9):24-28, 1998

[5] Lum Jia Jun, Brandon, "*Implementing Zero-Knowledge Authentication with Zero Knowledge (ZKA_wzk)*", the Python Papers Monograph Proceedings of PyCon Asia-Pacific 2010. Vol. 5, issue 3, p1

[6] Ijayan Jaikumar. "*Web Application Security Is Growing Problem for Enterprises*,http://www.infoworld.com/d/securitycentral/webapplications ecurity-growing-problem-enterprises-843, last visited 1/5/2013.

[7] Mahmood Khalel Ibrahem, "*Modification of Diffie–Hellman Key Exchange Algorithm for Zero Knowledge Proof*", ICFCN' 12, conf. baghdad, 2012.

[8] E. Trichkova, "*Application of PHP and MySQL for Search and Retrieval Web Services in Web Information Systems*" Proceedings of First International Conference on Information Systems & Datagrids, Sofia, Bulgaria, February 2005.

[9] Forouzan, Behrouz A., "*Cryptography and Network Security",* McGraw Hill, Int. Ed. 2008.

[10] Back, Amanda, "*The Diffe-Hellman Key Exchange",* http://129.81.170.14/~erowland/courses/2009-2/projects/Back.pdf, December 2, 2009.