A Novel Methodology for Secure Communications and Prevention of Forgery Attacks

M.V.Kishore,
M.Tech
Asst.prosessor
Dept of IT,
ANITS

G.Pandit Samuel,M.Tech
Dept of IT,
ANITS
Asst.prosessor

N.Aditya Sundar
Asst.prosessor
Dept of IT,
ANITS

M.Enayath Ali
(IT-IV/IV)
ANITS

Y.Lalitha Varma
(IT-IV/IV)
ANITS

ABSTRACT
Today almost all organizations in the world are network-centric paradigm and to safeguard the data in a world where technology is advancing, systems are changing rapidly and information flows freely requires efficient secure channel at the endpoint. Security is the heart of IT revolution and more specifically user authentication and key establishment are the rudimentary services in secure communications. Though many systems, schemes bank on public key digital certificate user authentication and key establishment, failed in getting authenticated due to some forgery attacks. Public key Digital certificate though gained popularity in the public key infrastructure (PKI) in providing authentication to user public key, itself cannot be used to safeguard an authenticate user. In this paper, we propose a novel approach using GDC for user authentication and key establishment. A GDC is a kind of Digital Certificate which contains user's public information and Digital signature which is issued and signed by the trusted Certificate Authority. The advantage of GDC is that, unlike the public key Digital Certificate, it does not contain user's public key. So, the digital signature can never be revealed to the verifier and this is where a digital signature of GDC becomes a security factor that can be used for user authentication. Using this phenomenon, we have implemented a Discrete Logarithm Protocol which satisfies in achieving user authentication and secret key establishment. In addition to this, by using the shared-secret key, we have also exchanged the data between the entities through AES (Advanced Encryption Standard) Cryptographic algorithm.

Keywords: Generalized digital certificate, user authentication, key establishment, shared-secret key, forgery attacks, data exchange(encryption and decryption).

1. INTRODUCTION
The main objective of cryptography is to provide security to data that has to be transferred through a secure Public Network. The major principles of security like authentication, confidentiality, Data Integrity, Non-repudiation are being agreed on the current public key Infrastructure model(PKI model). A public key infrastructure (PKI) is a system of digital certificates, certification authorities (CAs), and registration authorities that verify and authenticate the validity of each entity that is involved in an electronic transaction through the use of public key cryptography. A Digital Certificate consists of a statement and the digital signature of the statement, that is signed by trusted certificate Authority who issues certificates to each registered user. The statement in a Digital Certificate consists of Public key which proves the uniqueness of the user with respect to the trusted CA. The current system of PKI consists of X.509 Digital Certificate[3], that has been widely using in authentication of a user. The authentication of a user in X.509 Digital Certificate can be accepted if he is able to prove that he has knowledge of private key corresponding to the public key mentioned in the certificate.

In X.509 Digital Certificate, the sender's digital signature is encrypted with the receiver's public key. So, when the receiver gets the encrypted digital signature of the sender with his public key, he uses his private key to decrypt it so as to get the Digital signature of the sender. Then the receiver decrypts the digital signature of the sender by applying sender's public key which he got through the Digital Certificate of the sender to get the original text he received. Even though this mechanism works well to satisfy the basic security principles, there is a flaw identified in this that is, there may be a chance of forgery attack on the sender's Digital Signature. The Digital signature of the sender can be captured in the communication as when the receiver decrypts the whole text with his own private key. If a malicious receiver got the digital signature and can encrypt it with the third party's public key to send it in the name of legitimate sender. Then the third party will think that this message came from the legitimate sender only. This forgery attack is not addressed in the current public key Digital Certificate. To prevent this attack, we propose a novel design by using Generalized digital Certificate(GDC) for user authentication and key establishment[1]. This Generalized Digital Certificate does not contain any user's public key, instead it maintains user's public information such as SSN number, Digital driving license etc. And the sender's digital signature is also not sent to the verifier (receiver) directly in plain text. To prove his genuintity the sender has to face a challenge provided by the verifier. If the sender is able to prove his genuinity by responding to the verifier's challenge, then the user is authenticated. For this authentication mechanism we have implemented a new Discrete Logarithm based protocol. However, in this protocol there is no public key and no digital signature consent, so the malicious receiver cannot forge the digital signature of the sender.

In addition to this, the DL based protocol also involves in establishing a shared session secret key, which can be used for user encryption and Decryption of the messages for secure communica-tions[2]. Since, this shared key is just one time session key, even though the intruder gets the shared key, he cannot use it for other sessions. However getting the shared key is difficult task to an intruder since it can only be generated by applying random integers from the two entities(i.e the user and the verifier uses different key pairs for generating the shared session key). Since there is no specific

public-private key pair for GDC, the key management is very easy than public key digital certificate.

The following are the related works done to implement DL based protocol using GDC. Apart from GDC providing user authentication and key establishment, there are some public key authentica-tion based protocols available which rely on Public Key Digital certificates.

Undeniable Signature:
Undeniable signature scheme is a non-self-authenticating signature schemes where signatures can only be verified with the signer's consent. However, if a signature is only verifiable with the aid of a signer, a dishonest signer may refuse to authenticate a genuine document. Undeniable signatures solve this problem by adding a new component called the disavowal protocol in addition to the normal components of signature and verification. The scheme is implemented using public-key cryptography based on the discrete logarithm problem[4]. Verification is carried out by a challenge-response protocol where the verifier sends a challenge to the signer and views the answer to verify the signature.

ElGamal Digital Signature
The ElGamal signature scheme is a digital signature scheme which is based on the difficulty of computing discrete logarithms. It was described by Taher ElGamal in 1984. The ElGamal signature scheme allows a third-party to confirm the authenticity of a message sent over an insecure channel.

Message Digest (MD5):
MD5 algorithm takes input message of arbitrary length and generates 128-bit long output hash[5]. These generation functions must fulfill these requirements:

1) No one should be able to produce two different inputs for which the transformation function returns the same output

2) No one should be able to produce input for given pre specified output.

Digital Certificate:
In cryptography, a digital certificate or identity certificate is an electronic document that uses a digital signature to bind a public key with an identity — information such as the name of a person or an organization, their address, and so forth. The certificate can be used to verify that a public key belongs to an individual.

DL (Discrete Algorithm)
The Discrete logarithms[8] are the group-theoretic analogue of ordinary logarithms, which solve the same equation for some real numbers b and g, where b is the base of the logarithm and g is the value whose logarithm is being taken. In view of mathematics, a discrete logarithm defines an integer k solving an equation b^k = g, where b and g are elements of a group. Computing discrete logarithms is believed to be difficult. No efficient general method for computing discrete logarithms on conventional computers is known, and several important algorithms in public-key cryptography base their security on the assumption that the discrete logarithm problem has no efficient solution.

2. METHODOLOGY

1) Implementation of DL-based Protocol:

a) Prologue

There is a pitfall Public-Key Digital Certificate that is it alone cannot safeguard and provide authentication to user's public key due to some forgery attacks and manipulation of digital signatures.

To overcome this drawback, in our scheme we have proposed a novel approach using GDC for user authentication and key establishment. In GDC, digital signature becomes the major security factor that can be used for user authentication as it can never be revealed to the verifier.

In this paper, the implementation of DL-based based protocol is a blend of DL-based digital signature for user authentication and non-repudiation and the Diffie-Hellman key agreement protocol for obtaining the shared session key.

b) Overview of Elgamal Digital Signature, Hashing functions and Diffie-Hellamn Key Agreement Protocol

The ElGamal signature scheme is a digital signature scheme[1],[6],[7] which is based on the difficulty of computing discrete logarithms. The ElGamal signature scheme allows a third-party to confirm the authenticity of a message sent over an insecure channel. ElGamal signature is usually generated by the CA and it consists of $(r_A, s_{A)}$ where 'r' represents the random integer and 's' represents the signature component (secret key) assigned by the CA to the user. This signature in our protocol can be verified by the verifier to authenticate the user's message digest 'm' by checking whether the following equation

$$\mathbf{g^m = y^r r^s \bmod p},$$

holds true.

The role of Hashing function in our protocol is mainly used for generating the shared-secret key which is just a one-time session key and is generated by applying random integers from the two entities (i.e the user and the verifier's key pairs). A cryptographic hash function is a hash function that takes an arbitrary block of data and returns a fixed-size bit string, the cryptographic hash value such that any change to the data will change the hash value. The hash value is sometimes called the **message digest**.

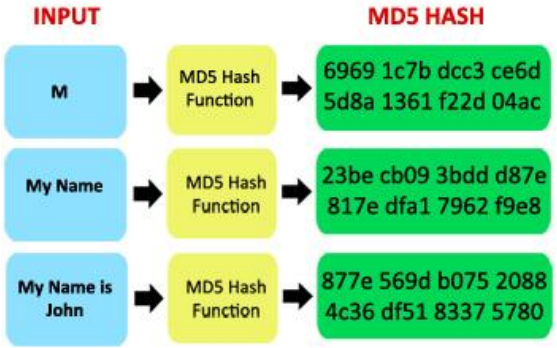

Fig 1. A cryptographic Hash Function.

The role of Diffie-Hellman Assumption in our protocol is to generate a one-time session shared-secret key which is used for exchanging the data between the entities for secure communications. By using the user's (A) and the verifier's

(B) private keys x_A and x_B, and their corresponding public keys, y_A and y_B. Only user and verifier can calculate a shared-secret key $K_{A,B} = y_B^{x_A} = y_A^{x_B} = K_{B,A} \bmod p$.

Diffie-Hellman protocol is based on the assumption that it is mathematically impossible to determine $K_{A,B}$ without knowing the private key x_A or x_B .

c) Mechanism of DL-based Protocol

Inorder to achieve user authentication and key establishment along with some additional security requirements like Unforgeability, One-wayness, Non-repudiation then it is possible through DL-based protocol. Following are the entities involved in the implementation of DL-based protocol:

1) **User:** The user who is the owner of GDC, is a person who receives the GDC from a trusted CA over a secure channel. The owner proves to the verifier's challenge question that he has the knowledge of the signature.

2) **Verifier:** The verifier is the person who challenges the owner of the GDC with a "question" and validates the answer using the user's public information, message digest (m) sent and CA's public key.

The steps involved in the mechanism of DL-based protocol are as follows:

a) Initially user sends the message digest m_A along with the ElGamal Signature (r_A ,s_A) to the verifier.
b) Verifier verifies for user authentication by computing the values send by the user through this equation:
$g^m = y^r r^s \bmod p$.
c) If the equation holds true then the user is authenticated and authentication is success else authentication fails.
d) On successful authentication the verifier randomly selects v_B and computes $c_B = r_A^{v_B} \bmod p$ as a challenge question and sends it to the user.
e) Now user using the Diffie-Hellman Key Agreement Protocol randomly selects v_A and computes $c_A = r_A^{v_A} \bmod p$ along with $K_{A,B}$ and ACK and sends it to the verifier.
f) After receiving the ACK and c_A from the user, the verifier uses his secret key v_B to the Diffie-Hellman shared-secret key and checks whether the ACK is true. If the verification is successful, then the user is authenticated by the verifier and a one-time secret-session key is generated between both the entities which provides an efficient path for security.

The below mentioned figure is used for better and simple understanding of the above mentioned steps. This figure step by step explains the working of DL-based protocol along with participation of entities, user authentication and key establishment.

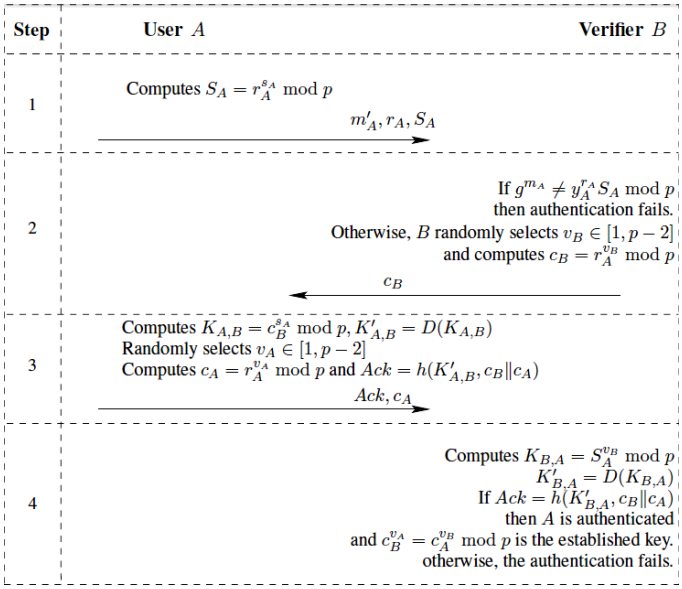

Fig 2 Implementation of DL- based protocol for user authentication and key establishment

d) Justification of Security Analysis

It is clear from the previous section that the proposed protocol satisfies the security- requirements: Unforgeability, One-wayness, Non-transferrability[1] and Non-repudiation. In this section, we are going to analyze how these security properties are satisfying to the DL-based protocol in achieving user authentication and key establishment.

1. **Unforgeability:** It is infeasible for the intruder to perform a forgery attack without knowing the secret signature component s_A to get a pair (r_A ,s_A) to satisfy it in the equation $\mathbf{g^m = y^r r^s \bmod p}$ and also compute $K_{A,B}$ to forge a valid ACK. Thus, the DL-based protocol satisfies its security against forgery attacks.

2. **One-wayness:** It is infeasible to compute secret key s_A from the signature component S_A because it is a DL-based protocol problem and even though the verifier knows the Diffie-Hellman key $K_{A,B}$ which is used by the user in computing the $K_{A,B}$, but the verifier cannot obtain the secret key s_A as it is never revealed during by the user during their communication. Thus, the DL-based protocol satisfies its one-wayness property.

3. **Nontransferrability:** Since we are using the concept of Diffie-Hellman Key Agreement Protocol in DL-based protocol, a valid response ACK can only be generated by a user knows who his secret digital signature component s_A issued by the CA, or by a verifier who knows his random challenge question c_B selected randomly to validate the user, and this response ACK is valid for only one-time session. Thus, there is no chance of unauthorized user to pretend as authorized to the opponent entity. Thus, the DL-based protocol also satisfies its nontransferrability property. Since, the digital signature of the user is never revealed during the communication and there is no chance of it getting manipulated as there is no privacy intrusion problem the proposed protocol also satisfies the non-repudiation property.

2) Data Exchange using AES Cryptographic Algorithm

In this paper, we have also embedded an additional feature of providing security by extending the former GDC paper. Once the one-time secret session key is obtained which is based on the Diffie-Hellman Assumption, we have successfully exchanged the data between the entities using the secured Advanced Encryption Standard (AES) Cryptographic algorithm.

Overview of AES Cryptographic Algorithm

AES is a variant of Rijndael which has affixed block size of 128 bits, and a variable key size of 128,192 or 256 bits. AES operates on 4x4 column-major order matrix of bytes, termed the 'state', although some versions of Rijndael have a larger block size and have additional columns in the state. Most of the AES calculations are particularly done in a special finite field. The key–size used for the AES cipher specifies the number of cycles or transformations that convert the input, which is the "plaintext", into the final output, called the "cipher text".

For example:

10 cycles of repetition for 128-bit keys.

12 cycles of repetition for 192-bit keys.

14 cycles of repetition for 256-bit keys.

Steps in AES algorithm

Initial Round

1) Add Round Key: Each byte of the state is combined with a block of round key using Bitwise Xor.

Sequence of Rounds

1) Sub-Bytes: It is non-linear substitution step where each byte is replaced with another according to a lookup table.
2) Shiftrows: It is a transposition state where the last three rows of the state are shifted cyclically to a certain number of steps.
3) MixColumns: It is a mixing operation which operates on the columns of the state, combining the four bytes in each columns.
4) Add Round Key

Final Round

1) Sub-Bytes
2) Shift rows
3) Mix Columns

Internal Operation Of AES Algorithm

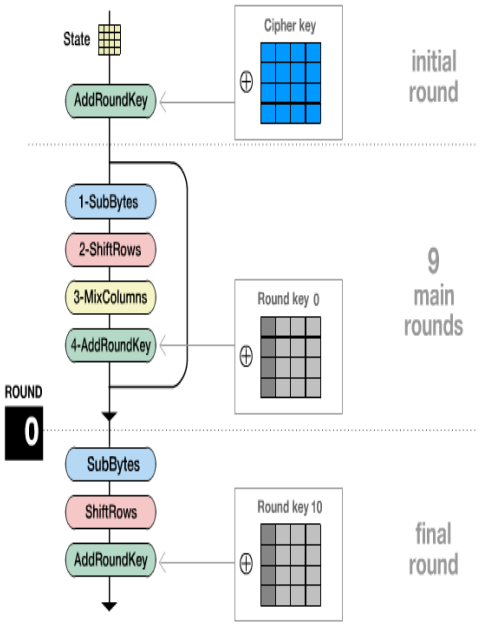

Fig 3 Working of AES Algorithm

Advantages of AES algorithm

1) It has a variable 3Key pair 128,192,256 size.
2) It is much faster than 3DES(5-10 times faster) in terms of performance.
3) It is complex to crack (decrypt) the text using AES algorithm.

3. IMPLEMENTATION AND DATA ANALYSIS

The practical implementation of the topics, mentioned under methodology are explained in this section with detailed description. As per data analysis for DL-based protocol, it contains the following modules:

1) User Module
2) Verifier Module
3) Authentication Module
4) Key Generation
5) Data Exchange Module using shared-secret

User module: Once the Client and Server are connected the User window is invoked by allowing the user to participate in the communication by entering the values.

User window

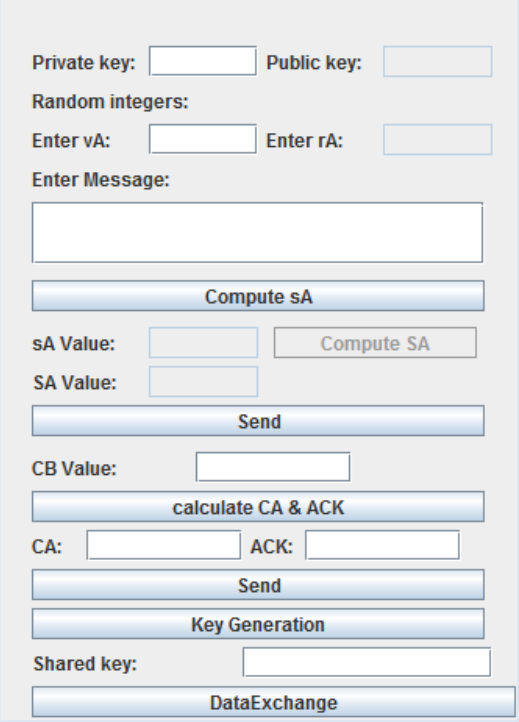

Fig 4 User-side Window

Verifier module: Similarly, in this module also once the Client and Server are connected the verifier window is invoked by allowing the verifier to participate in the communication.

Verifier window

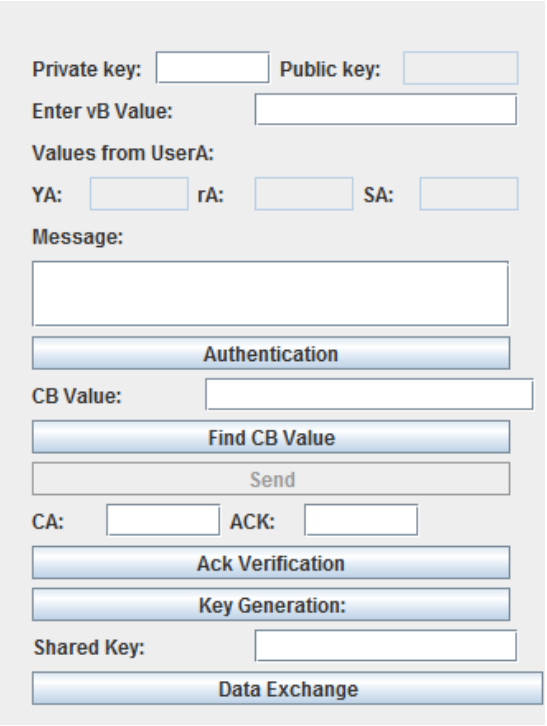

Fig 5 Verifier–side Window

Authentication module: Once the user window is invoked the user begins the communication with the the ElGamal Signature (r_A , s_A) to the verifier.

The verifier verifies for user authentication by computing the values send by the user through this equation:

verifier by sending the message digest m_A along with

$$g^m = y^r r^s \bmod p.$$

If the equation holds true then the user is authenticated and authentication is success else authentication fails.

User window

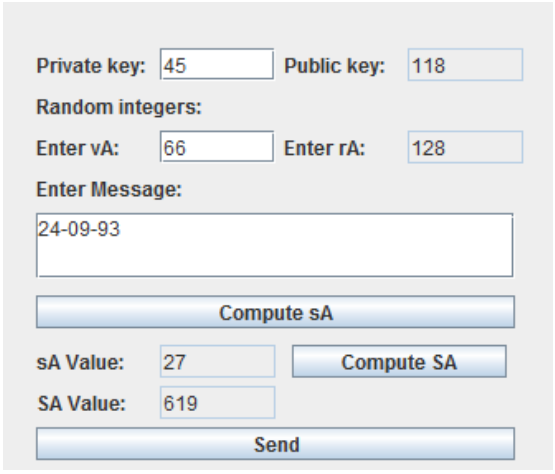

Fig 6 User computing S_A and Message digest m for user authentication

Verifier window

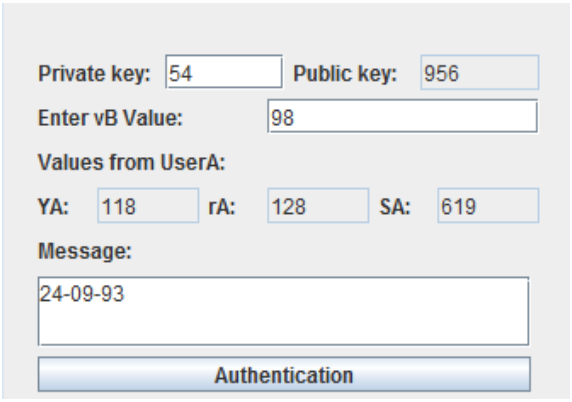

Fig 6.1 Verifier validating the respective user

Mathematical analysis of the equation, $g^m = y^r r^s \bmod p$ and the user authentication based on the values sent by the user is explained as follows:

Initially, the user selects a large prime 'p' and a generator 'g' in the order of p-1 which are shared and agreed commonly by all the entities. Now the user selects a random private key p1 € [1,p-1] and basing on it computes the corresponding public key $y_A=g^k \bmod p$. Then the user randomly selects a secret

value (parameter) k € [1,p-1] with gcd (k,p-1)=1 and computes random integer $r_A=g^k$ mod p.

From the above figure at user side, the user enters the random private key 'p1' as 45 and basing on it the public key y_A, random integer r_A are obtained respectively as 118, 128. Now the user calculates 's_A' (secret key) by using this formula m=ks + rx mod p-1 and from the figure we get the value as 27. After getting the s_A value, the user calculates the 'S_A' such that $S_A= r_A^{s}{}_A$ and we get the value as 619. Now the user substitutes these values in this equation:

$$\mathbf{g^m = y^r r^s \bmod p.}$$

Here 'g' is assumed to be 122 and 'p' is assumed to be 123 and 'm' is assumed to be 3. Thus we get the $\mathbf{g^m}$ value as 1815848 and similarly on substituting the values of 'y_A', 'r_A', 'p' on the right-hand-side we get 1.77529*100 ≈181529. Hence both the sides of the equation are obtained as approximately equal. Now the user sends the ElGamal signature (r_A , s_A), message digest 'm' and public key y_A to the verifier. On receiving these values from the user, the verifier also substitutes these values in this equation:

$$\mathbf{g^m = y^r r^s \bmod p.}$$

The verifier also gets both the sides of the equation equal as 'g' and 'p' values are commonly shared and agreed by both the entities. Thus, at both the ends the above equation is proved and satisfied as per the mathematical analysis and basing on this identity the user is authenticated by the verifier.

Key Generation module: Once the user is authenticated the verifier randomly selects v_B and computes $c_B=r_A^{v}{}_B$ mod p as a challenge question and sends it to the user for the purpose of non-repudiation.

Verifier window

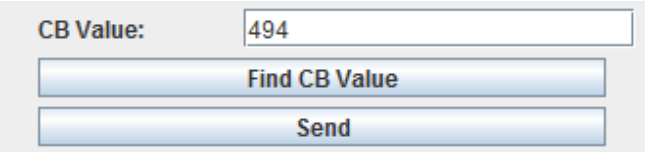

Fig 7 Verifier Calculating c_B value

Now after getting the c_B value from the verifier, the user using the Diffie-Hellman Key Agreement Protocol randomly selects v_A and computes $c_A= r_A^{v}{}_A$ mod p and response ACK and sends it back to the verifier.

User window

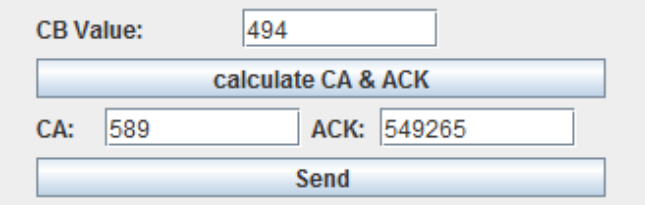

Fig 7.1 User calculating c_A and ACK

After receiving the ACK and c_A from the user, the verifier uses his secret key v_B to the Diffie-Hellman shared-secret key and checks whether the ACK is true. If the ACK is true, the authentication is successful else fail.

Verifier window

Fig 7.2 Verifier checking for ACK verification

The mathematical analysis of the Key Generation module is explained as follows:

Once the user is authenticated the verifier computes the c_B value by substituting the values of r_A , $^{v}{}_B$ in this equation: $c_B=r_A^{v}{}_B$ and gets the approximate value as 493.778 ≈ 494 and sends it to the user.

The user on receiving the c_B value, uses his secret key s_A to compute the Diffie-Hellman secret-key $K_{A,B} = c_B^{s}{}_A$ mod p. Now the user computes the c_A value by substituting the values of r_A , $^{v}{}_A$ in this equation: $c_A= r_A^{v}{}_A$ mod p and gets the approximate value as 589. Along with c_A the user also calculates the response ACK=h($K_{A,B}$, c_B‖ c_A), where 'h' represents a one-way hash function. From the above figure the 'h' value i.e the ACK value is obtained as 549265. Now the user sends the c_A and the response ACK values to the verifier.

The verifier on receiving the ACK and c_A from the user, the verifier uses his secret key v_B to the Diffie-Hellman shared-secret key and checks whether the ACK is true i.e the verifier solves the equation $K_{B,A}= S_A{}^{v}{}_B$ mod p and checks whether h($K_{A,B}$, c_B‖ c_A)= ACK is true. Thus, on substituting the values of S_A, $^{v}{}_B$ in the above in this equation: $K_{B,A}= S_A{}^{v}{}_B$ mod p, the verifier also gets the same value 549265, which is sent by the user. Thus, using the Diffie-Hellman protocol, the response ACK is same on both the sides and holds true for the hash function h($K_{A,B}$, c_B‖ c_A)= ACK which clearly specifies the successful ACK verification.

If the verification is successful, then the user is authenticated by the verifier and a one-time secret-session key is generated between both the entities which provides an efficient path for security. The user and the verifier clicks on the **Key Generation** button to generate shared-secret key for the purpose of key agreement and secure communication.

User window , Verifier window

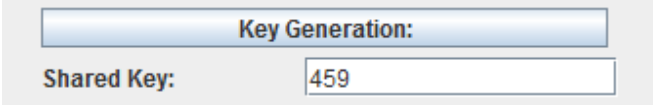

Fig 7.3 Shared-secret Key generated at both sides (user,verifier)

Data Exchange module using shared-secret key: Now after getting this One-time shared session key we have also provided an additional facility for exchanging the data between both the entities (user,verifier) using a secured AES cryptographic algorithm.

The user and the verifier clicks on the **Data Exchange** button in their respective windows for exchanging the data in a secured fashion. After clicking the Data-Exchange button 2 windows are invoked simultaneously at both the ends allowing both the entities to transfer their data basing on the shared-secret key.

User window , Verifier window

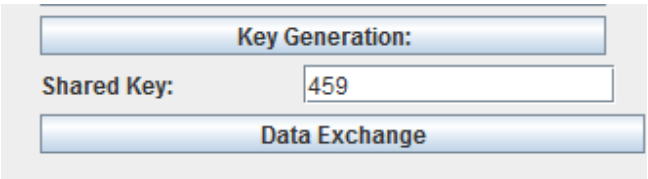

Fig 8 Data Exchange button for exchanging of data

The data exchange between the entities happens by undergoing the internal operation of AES and MD-5 algorithm such that, at the user side the user enters the data to be sent into the text field and clicks on the **Encrypt and send** button to maintain the confidentiality and integrity of the message.

The verifier on receiving the data clicks on the **Decrypt and receive** button to decode the cipher text into the original text, once the original message is received by the verifier, to send the data to the user the verifier also follows the same procedure by entering the text into the text field and clicking the encrypt & send button. In this way, the user also receives the encrypted message from the verifier. This entire mechanism is described below in this diagram:

User window

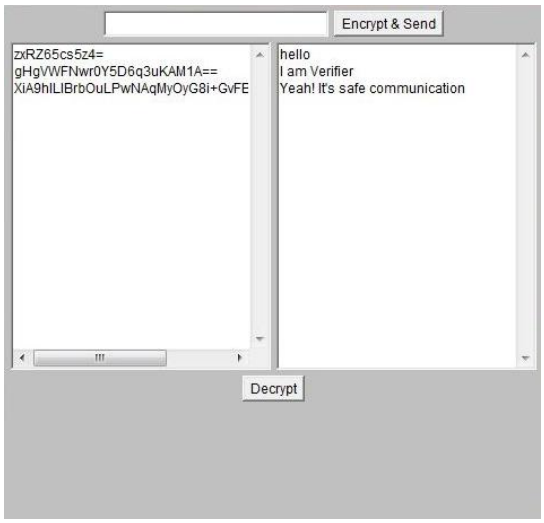

Fig 8.1 Data transmitting and receiving at User-side

Verifier window

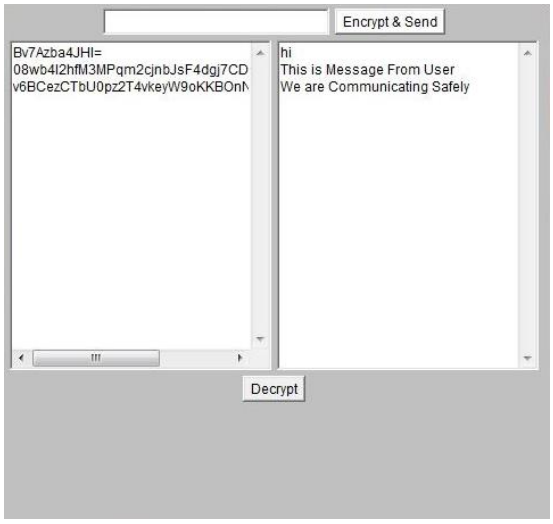

Fig 8.2 Data transmitting and receiving at Verifier-side

4. CONCLUSION AND FUTURE ENHANCEMENT

In this paper, we have proposed GDC as a novel approach by highlighting its salient features for user authentication and key establishment when compared to the existing X.509 Public Key Digital Certificate. We have also implemented the DL-based protocol based on the concept of GDC, which works successfully and efficiently by generating optimized values at each and every module. We have also embedded an additional facility in this paper by providing the data exchange module based on one-time session key generated by using the secured AES cryptographic algorithm.

For future enhancement, the proposed system can be extended by implementing another protocol which is more secured, efficient, feasible when compared to the DL-based protocol. The another alternative is the IF- (Integer Factor) based protocol whose security depends on the combination of RSA Signature and One-way- hash function. The protocol provides deniable authentication and protects privacy of the digital

certificate. In this paper, we have used AES algorithm for secure exchange of data between the entities for further extension our approach can be applied to more advanced and secured cryptographic techniques for secure exchange of data.

5. REFERENCES

[1] L.Harn and J.Ren, "Generalized Digital Certificate for User Authentication and Key Establishment for Secure Communications", IEEE trans. on Wireless Communications, vol.10,pp.2372-2379,2011.

[2] Bismin.V.Sherif and Andrews Jose, "Secure Communication using Generalized Digital Certificate", International Journal of Computer Applications Technology and Research, Volume 2-Issue 4,396-399,2013.

[3] Network Working Group, "Internet X.509 Public key Infrastructure Certificate and crl profile, RFC:2459," Jan 1999.

[4] D.Chaum and H. van Antwerpen, "Undeniable Signatures," Advances in Cryptology-Crpyto'89, Lecture Notes in Computer Science, vol. 435, pp. 212-217, 1989.

[5] R.Rivest, A.Shamir, L.Adleman, "A method for obtaining Digital Signatures and Public-Key Cryptosystems," Commun. Assoc. Comp. Mach., Vol. 21, no. 2, pp. 120-126, 1978.

[6] T.A. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," IEEE. Trans. Inf. Theory, vol. 30, no. 24,pp. 469-42, 1985.

[7] L. Harn and Y.Xu, "Design of generalized ElGamal type digital signature schemes based on discrete logarithm, "Electron. Lett., vol. 30, no. 24.pp. 2025-2026, 1994.

[8] en.wikipedia.org/wiki/Discrete-logarithm