Advances in Computer Science and its Applications (ISSN: 2166-2924)
Vol. 1, No. 1, March 2012

www.worldsciencepublisher.org

Message Level Security Realization in Web Services Using AES and Diffie Hellman Key Exchange

[1]Priyadharshini M, [2]Sneha Raichel Mathew, [3]Baskaran R, [4]Suganya.V

[1,2,3,4] Department of CSE, Anna University, Chennai, India

Email: mpriya1977@gmail.com , sneharmathew@gmail.com , suganya@cs.annauniv.edu , baaski@cs.annauniv.edu

Abstract- Major usage of Internet elevates the significance of web services, which in turn makes web service security a very challenging issue. Web Service uses SOAP to exchange information; although SOAP guarantees XML security, XML is still liable to attacks like XML rewriting, XML bombing, external entry attack, denial of service etc. Hence providing security at the message level seems to be important with respect to web services. This proposed work provides a method to enhance the security of the web service at message level by encrypting the SOAP message using AES, with the help of shared key generated using Diffie Hellman key exchange mechanism. The key exchange is implemented as service and a digital signature handler is provided to enable a secured key exchange and is done well before the SOAP message generation. The main feature of this proposed system is that the variable keys are used for encryption each time the request is sent which prevents hacking of messages in application invoking web services.

Key Words-Web Service Security; SOAP message; AES; Diffie Hellman; XML Security.

1. Introduction:

Web service [1] is an application stored in one machine that can be accessed by another machine over a heterogeneous and a geographical distributed network. The application that accesses the web service sends a method call to the remote machine, which processes the call and sends a response to the application. Web service messages are sent using SOAP. SOAP structure is

```
<? xml version=”1.0” encoding=”UTF-8” >
<HEAD>
    <SID>Session Id</SID>
    <TIMESTAMP>timestamp</TIMESTAMP>
    <IP>IP address</IP>
</HEAD>
<BODY>Soap Message body</BODY>
<HASH>hash</HASH>
```

The message that is being sent is kept inside the SOAP BODY enclosed by the function parameter. In spite of the tighten security given to the message in the SOAP, it is still liable to XML rewriting attacks [2]. Thus the message sent within the SOAP is not secure.

Cryptography is a technique of converting a message into a form understandable only by the sender and receiver. Incorporating cryptographic techniques into the SOAP message will grant a secure web service. Among the various cryptographic techniques encryption is widely used. Encryption can secure the data by converting the given data to a cipher by using an encryption key and an encryption algorithm. To decrypt the cipher one should have access to the key. Encryption provides privacy, confidentiality, secrecy, non-repudiation, availability, reliability and authentication when communicated through an unsecured channel [3].

There are mainly two types of encryption algorithms. They are symmetric key encryption algorithm (same key used for encryption and decryption) and asymmetric key encryption algorithm (different key used for encryption and decryption) [3].

Asymmetric key encryption provides confidentiality and authentication [3]. The various symmetric key encryption algorithm, are DES, 3DES, BlowFish and AES. DES algorithm was introduced during1974 ever since then it was prone to attack. To increase the complexity of DES, 3DES was introduced; 3DES is comparatively slower than other encryption techniques. Blow Fish on the other hand has good performance but is prone to attack [3, 7, 9]. AES is a relatively new and found to be one of the effective encryption techniques since it is strong against differential, linear, interpolation and square attack.

An attack is usually centred to mathematics used in the algorithm (i.e., differential and linear cryptanalysis). An algorithm can be secured only if its hardware is also secure. The weakness of the algorithm’s hardware can cause it to leak out critical information. These attacks are called side channel attack which also includes timing attacks, simple power analysis and differential power analysis [4]. By these attacks the cryptanalysis aims at determining the secret key. The determination of the secret key in long run could be avoided if the secret key is generated just before the message exchange and the message will be secure. The message encryption is done using the secret key generated using the Diffie Hellman key exchange method [8] and hence the secret key is only know by the receiver and sender. Diffe Hellman Key exchange could also be prone to man in the middle attack which could again be resolved by introducing a Digital Signature to ensure the sender’s authenticity [6].

2. Background

To say web service as secure, it should be authorized, authenticated, confidential, anti-denial and data integrity. Authentication is establishment of proof of identities among entities involved in the system.[my] Authorised means the message is send and received by the person with proper privileges. Confidential means that the information is

known only by the sender and the receiver. Anti-denial ensures that a message send can't be denied. Data integrity means the data received is not changed during transit. To ensure security there are many standards that are proposed by Organization for the Advancement of Structured Information Standards (OASIS) and World Wide Web Consortium (W3C). All these standards give a provision to use the security mechanisms, which sometimes could be customised to increase the level of security.

Till now DES was often used and rarely 3DES was used to secure SOAP message to ensure integrity. Hence in this paper we try to propose a customisation to secure the SOAP message using AES. Considering both the service provider and the service requestor have their respective public key-private key pair and the others public key, a shared key is generated using the Diffie Hellman approach and exchange is done by signing it using digital signature. Then encrypt the SOAP message using AES with the help of key generated using Diffie Hellman and sent as request to the service provider. The SOAP is received at the service provider; the encrypted message is then decrypted using AES. The original message is obtained and the service is executed to provide the response to the service requestor.

Advance Encryption Standard (AES) algorithm a symmetric key encryption technique was developed in 2000. Selection of the AES depends on the key size. AES divides the message into 128 bit data block that is supported by key size of 128, 192 and 256 bit. The number of rounds in AES is depended on the key size (i.e. AES128 10 rounds, AES192 12 rounds AES256 14 rounds). Working of AES can be divided into two halves Key Expansion and Message Encryption. Key Expansion phase expands the key into the required number of keys. In Message encryption phase each data block goes through n rounds depending on the key size. Each round consists of a SubByte substitution, ShiftRow, MixColumn and Add round key.

Diffie Hellman algorithm introduced by Whitfield Diffie and Martin Hellman in 1976 using the concept of discrete logarithm. It was the first system to utilize 'public-key' or 'asymmetric' cryptographic keys [5, 6, 8]. This algorithm gives the user full fledge advantage of not keeping a shared key. The key can be generated as and when required. The requestor generates a random number rrn and creates a public key $K_r = G^{rrn} \bmod M$ which is then sent to the provider. The provider selects another random number prn and generates its public key $K_p=G^{prn} \bmod M$. Then using the K_r and prn, the receiver generates a shared key $K_{sp}=K_r^{prn} \bmod M$ and sends K_p to the requestor. K_p is received at the requestor end where $K_{sr}= K_p^{rrn} \bmod M$ is generated. Both the keys are the same as

$K_{sr}= K_p^{rrn} \bmod M$
$= (G^{prn} \bmod M)^{rrn} \bmod M \; [K_p=G^{prn} \bmod M]$
$= G^{prn\ rrn} \bmod M$
$= G^{rrn\ prn} \bmod M$
$= (G^{rrn} \bmod M)^{prn} \bmod M$
$= K_r^{prn} \bmod M$
$= K_{sp}$

In section 3, the system architecture for the proposed system is discussed in detail with the process done by each component and sequence of interactions in the proposed system. In section 4 implementation details are provided and the inferences are recorded and discussed.

3. Proposed Architecture

The proposed architecture includes a set of components that are enhancements in the normal web service model to enable and encryption scheme so as to secure the SOAP message during the transit. The components that constitute the proposed system are discussed below in detail:

a) Key Management System: This component is responsible for generation of the public key -private key pair needed by the requestor and the provider. The basic functionality includes generation, storage, distribution, destruction etc., Requestor and Provider credentials are used to obtain the public key needed for digitally signing and verifying the keys transacted by the Diffie Hellman. This is used by the DSHandler for the above said purpose.

b) Client User Interface: The domain specific interface which is part of the client web application is designed and developed, from where the web services are invoked. This invocation causes the execution of services through the Security Handlers created. This is done once the needed service with appropriate security requirements has been discovered.

c) Generation Public Value (Requestor): The Service Requestor performs the following steps:
   a. Select a random private value rrn
   b. Using the rrn generates a Public value using $K_r=G^{rrn} \bmod M$
      where G and M are commonly shared parameters between the services.
   c. Generate signature for K_r, DSA (K_r, PR (requestor))
   d. Sends the Public key (K_r) with the Digital Signature generated to the provider
      K_r+DSA(K_r,PR(requestor))

   DS Handler component at the requestor side is involved in generation of Digital Signature for which key is provided by Key management system.

d) Generation Public Value (Provider): The Service Provider performs the following steps:
   a. Select a random no prn
   b. Using prn generates the Public key using $K_p = G^{prn} \bmod M$
   c. Generate signature for K_p, DSA (K_p, PR (provider))
   d. Sends the Public key (K_p) with the Digital Signature generated to the requestor
      K_p+DSA(K_p,PR(Provider))

   DS Handler component at the provider side functions same as that in the requestor side in generation of Digital Signature.

e) Generation Shared Key(Requestor):
   DSHandler component separates digital signature DSA(K_p,PR(provider)) and Public value K_p and verifies the digital signature using the PU(provider). Then the requestor generates Shared

key using the public value got from the provider K_p using

$K_{sr} = K_p^{rrn} \bmod M$

where G and M are commonly shared parameters between the service and the rrn generated earlier.

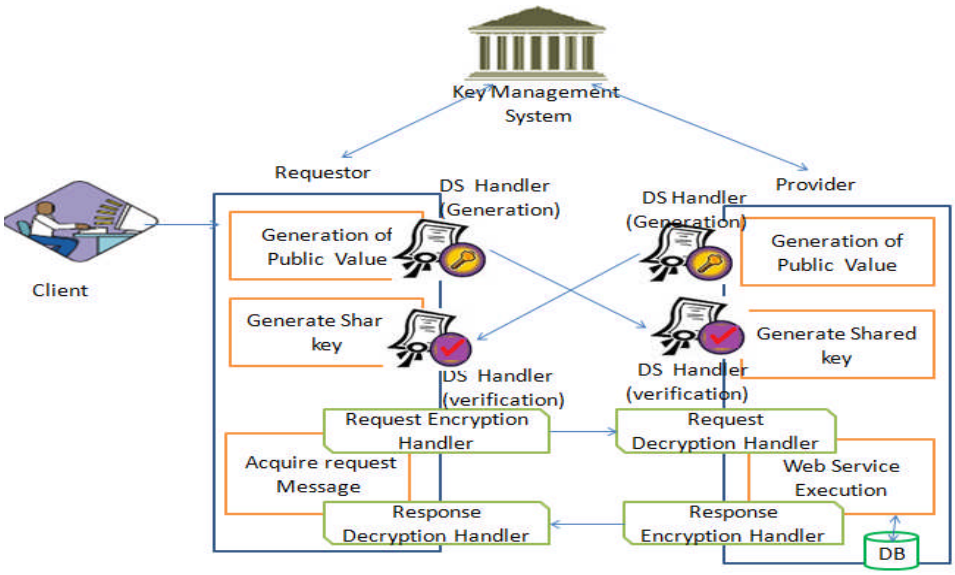

Figure 3.1 Enhanced Web Service Architecture Implementation

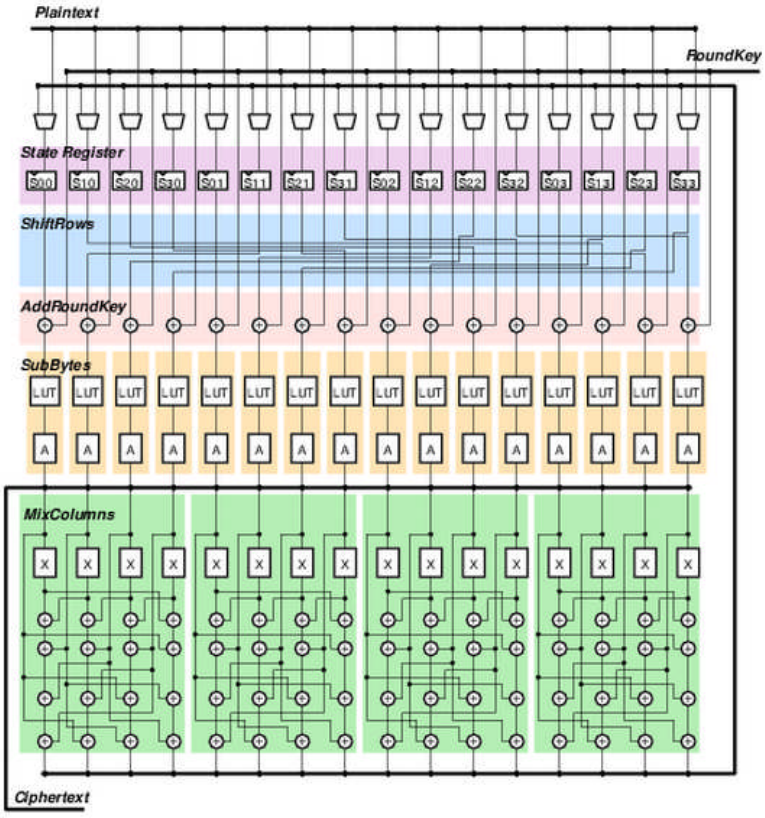

Figure 3.2. Design of AES encryption round

f) Generation Shared Key(Provider):

DSHandler component separates digital signature DSA(K_r,PR(requestor)) and Public value K_r and verifies the digital signature using the PU(requestor). Then the provider generates Shared key using the public value got from the provider K_r using

$K_{sp} = K_r^{prn} \bmod M$

where G and M are commonly shared parameters between the service and the prn generated earlier.

g) Request Encryption Handler: It is responsible for extracting the message from the SOAP request and the message acquired from the client is encrypted by AES using K_r. AESenc(K_{sr}, message) and is send to the Provider.
h) Request Decryption Handler: It is responsible for extracting the encrypted message from the SOAP request and decrypts it using AES algorithm with the generated secret key.
i) Response Encryption Handler: It is responsible for extracting the message from the SOAP response and encrypts it using AES algorithm with the generated secret key.
j) Response Decryption Handler: It is responsible for extracting the encrypted message from the SOAP response, AESenc(K_{sr}, message) received is decrypted using AESdec(K_{sp}, AESenc(K_{sr}, message))Since K_{sr}= $K_{sp.}$ It means AESdec(K_{sp}, AESenc(K_{sp}, message))
k) Web Service Execution: performs the requested service for the service requestor.

AES encryption and decryption is performed using handlers at service requestor and provider side as well. In AES the plain text (128 bit) is represented by 4X4 byte matrix. An AES round is composed of four operations:

1. SubBytes (SB),
2. ShiftRows (SR),
3. MixColumns (MC) and
4. AddRoundKey (AK).

The MixColumns operation is omitted in the last round and an initial key addition is performed before the first round. The number of rounds is variable depending on the key length, 10 rounds for 128-bit key, 12 for 192-bit key and 14 for 256-bit key.

The Encryption is done using the following algorithm:

```
encrypt( )
{
 Key K[1..14]=GenerateKey(Ksr)
 AddRoundKey(K[1])
 For (i=1;i<=14;i++)
    CipherText=GenerateCipher(K[i],PlainText)
SubByte(PlainText)
ShiftRow(PlainText)
MixColumn(PlainText)
}
GenerateCipher(K,PlainText)
{
SubByte(PlainText)
ShiftRow(PlainText)
MixColumn(PlainText)
AddRoundKey(K,PlainText)
}
```

Decryption is done using the following algorithm:

```
decrypt()
{
 Key K[1..14]=GenerateKey(Ksp)
 AddRoundKey(K[14],Plaintext)
 For(i=14;i>=1;i--)
    PlainText=GeneratePlainTxt(K[i], CipherText)
 InvSubByte(PlainText)
 InvShiftRow(PlainText)
}
GenerateCipher(K,PlainText)
{
 InvSubByte(PlainText)
 InvShiftRow(PlainText)
AddRoundKey(K,PlainText)
InvMixColumn(PlainText)
}
```

Since AES is a symmetric algorithm encrypting and decrypting with the same key will give back the message

The sequence of steps in the proposed system is as follows:

1. The client invokes the web service through the user interface.
2. Service Requestor selects a random private key value, generate a Public key and pass on to the Requestor DS Handler.
3. The Requestor DS Handler generates a digital signature using the private key of the requestor. Attaches this signature along with the public key and sends it as SOAP request to the provider.
4. The Provider DS Handler receives request. Extracts the message and the signature. Verifies the signature using the public key of the requestor (for integrity).
5. Service Provider obtains the public value the requestor from handler. Then select a random number and generates public key which is send to the provider.
6. Provider DS Handler extracts the message and generates a digital signature using the private key of the provider. Attaches this signature along with the message and sends it as response to the requestor.
7. The Requestor DS Handler receives the request. Extracts the message and the signature. Verifies the signature using the public key of the provider (for integrity).
8. Service requestor obtains the public value generates the shared and encrypts the message using AES algorithm with the help of encryption handler.
9. Service Provider will extract the message and decrypt the message using AES key with the help of decryption handler.
10. The service is then executed for the results, followed by encryption ant provider side and decryption at requestor side.

4. Implementation and Inferences

User Registration system which holds credit card information for booking tickets is designed and used as client application and that acts as the requestor. As the provider we have a banking system which could validate the credit card number. Since this card information is used for payment transaction this needs to be secured during transit for verification.

The interface information in client application is as in the figure shown here.

Figure 4.1 User interface

The credit card number is passed for validation on click of txtVeri button. The services designed for key exchange and card validation is as follows:

```
@WebMethod(operationName = "exChange")
public String exChange(@WebParam(name = "Publickey") String
Publickey) {
    key = Diffie.BASE_2.generateRandomKeys(modulus);
    a_shared_secret = Diffie.getSharedSecretKey(keys[0],
    modulus, Publickey);
    FileOutputStream output = new FileOutputStream(
    "secretkey.bin");
    output.write( a_shared_secret.toByteArray());
    return keys[1].toString();//Public key of the provider
}
```

```
@WebMethod(operationName = "ServicePer")
public String ServicePer(@WebParam(name = "EncMess") String
EncMess) {
  if( validator.ValidCreditCheck(EncMess))
     return("Valid card");
  else
     return("Invalid card");
}
```

The sequence of information passed during the service invocation involves handlers for Diffie Hellman key exchange as well as for AES encryption. The Diffie Hellman key exchange is performed in a secured way by digitally signing the key that is been exchanged. The output for the sequence of Diffie Hellman key exchange using Digital Signature, followed by AES using the key generated is represented as the output window using Figure 4.2.

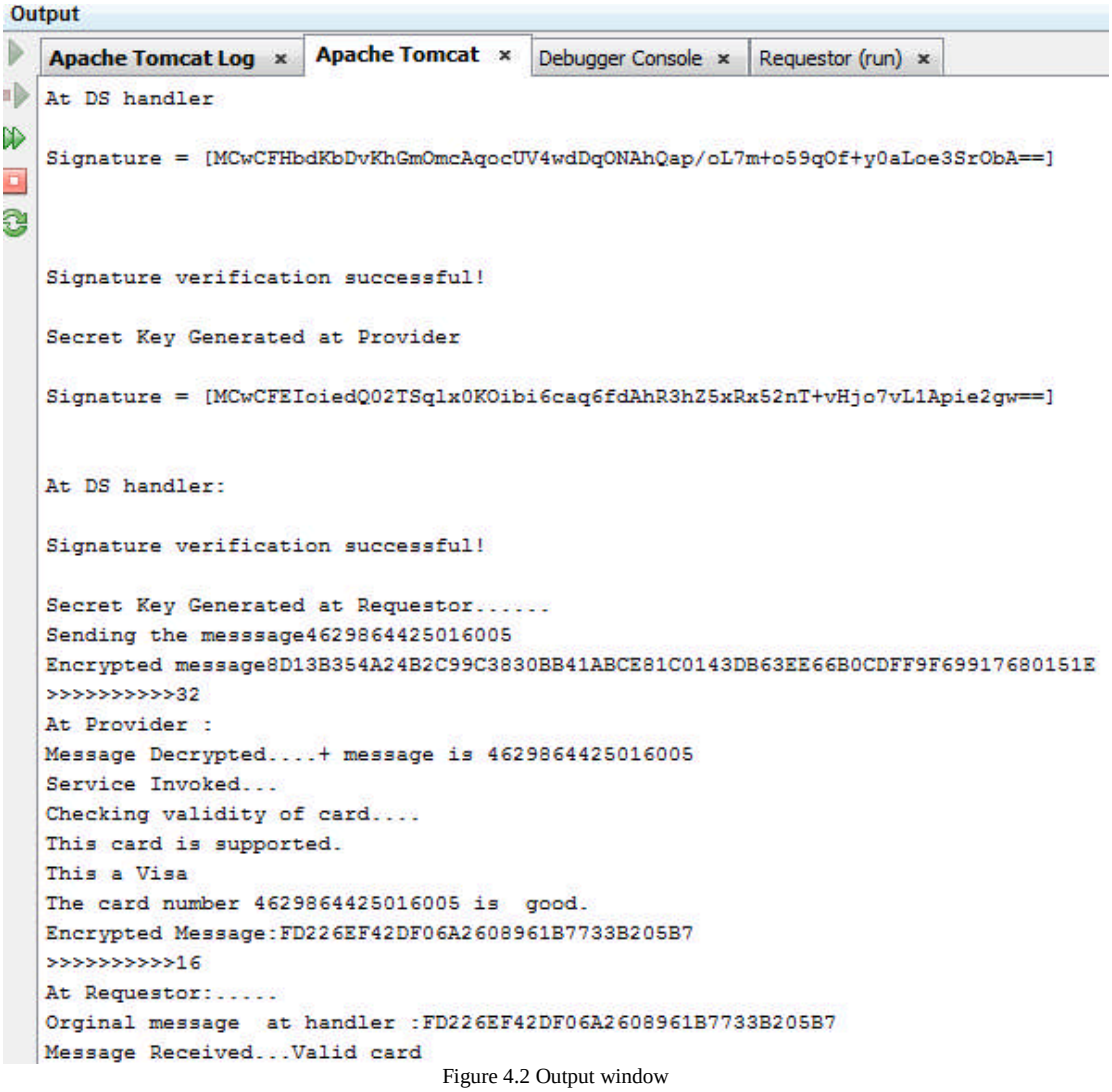

Figure 4.2 Output window

For AES algorithm the time required to check all possible keys at 50 billion keys per second for 128-bit key length is 5 X 10^{21} years. The complexity here is again increased by including the Diffie Hellman key exchange to introduce the variable keys and integrity is maintained using digital signature. The time taken by the system is recorded for AES with and without Diffie Hellman and digital signature for multiples runs.

Table 4.1 Time taken for AES with and without Diffie Hellman and Digital Signature.

Number of Invocations	Time taken AES (ns)	Time Taken AES with DH & DS(ns)
1	15	77
2	16	79
3	16	78
4	15	78
5	16	77

The values recorded could show the increase in time due to introduction of Diffie Hellman and digital signature which could increase the time taken to check all possible keys and break confidentiality and integrity of the key exchange.

5. Conclusion and Future Work

From the implementation and inferences obtained, we could ensure the confidentiality of the information during transit as well as integrity in the key exchange. Though any optimisation measure is not introduced to reduce the time taken to do the encryption and key exchange. This is a new enhancement not tired yet in the web service models. This could serve as a key to open up further more new developments in the area of message level security. By applying stronger decision Diffie Hellman assumptions could increase the security of systems using Diffie Hellman key exchange.

References

[1] Yann Le Blevec, Chirine Ghedira, Djamal Benslimane, Xavier Delatte and Zahi Jarir, 'Exposing Web Services to Business Partners: Security and Quality of Service Issue',2006,1st International Conference on Digital Information Management.

[2] K. Bhargavan, C. Fournet, A. Gordon, and G. O Shea, 'An Advisor for Web Services Security Policies', http://research.microsoft.com/~adg/Publications/details.htm#sws05

[3] Verma, O.P., Agarwal, R., Dafouti, D. and Tyagi, S.,'Performance Analysis of Data Encryption Algorithms', 2011, 3rd International Conference on Electronics Computer Technology (ICECT)

[4] Mohammad Zahiduf Rahaman and Mohammad Akram Hossain 'Side Channel Attack Prevention for AES Smart Card' Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008) 25-27 December, 2008, Khulna, Bangladesh

[5] Jie Liu and Jianhua Li 'A Better Improvement on the Integrated Diffe-Hellman-DSA Key Agreement Protocol' International Journal of Network Security, Vol.11, No.2, PP.114,117, Sept. 2010.

[6] Lein Harn, Manish Mehta, Student Member, IEEE, and Wen-Jung His 'Integrating Diffie–Hellman Key Exchange into the Digital Signature Algorithm (DSA)', IEEE Communication letters, vol. 8, No. 3, march 2004.

[7] Hamdan.O.Alanazi, B.B.Zaidan, A.A.Zaidan, Hamid A.Jalab, M.Shabbir and Y. Al-Nabhani "New Comparative Study Between DES, 3DES and AES within Nine Factors" Journal of computing, vol 2, issue 3 March 2010

[8] Whitfield Diffie and Martin E. Hellman 'New Directions in Cryptography Invited Paper', http://www.cfoworld.co.uk/ white-paper/networking /3619.

[9] Shashi Mehrotra Seth, Rajan Mishra 'Comparative Analysis Of Encryption Algorithms For Data Communication' IJCST Vol. 2, Issue 2, June 2011

[10] Yudong Zhang, Lenan Wu, "A Robust Hybrid Restarted Simulated Annealing Particle Swarm Optimization Technique", Advances in Computer Science and its Applications, Vol.1, No.1, pp. 5-8, 2012

[11] Iman Sadeghkhani, Ali Yazdekhasti, Arezoo Mortazavian, Nima Haratian, "Radial Basis Function Neural Network based Approach to Estimate Transformer Harmonic Overvoltages", Advances in Computer Science and its Applications, Vol.1, No.1, pp. 38-44, 2012

[12] Vikas Deshmane, "XML driven SCPI interpreter", Advances in Computer Science and its Applications, Vol.1, No.1, pp. 59-62, 2012