International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Special Issue 31(September, 2015), PP. 51-56

HYBRID RSA-AES ENCRYPTION FOR WEB SERVICES

AN IMPLEMENTATION OF HYBRID ENCRYPTION-DECRYPTION (RSA WITH AES AND SHA256) FOR USE IN DATA EXCHANGE BETWEEN CLIENT APPLICATIONS AND WEB SERVICES

Kalyani Ganesh Kadam
M.E. Information Technology Department
Terna College of Engineering
Navi Mumbai, India
kalyani.g.kadam@gmail.com

Prof. Vaishali Khairnar
H.O.D. Information Technology Department
Terna College of Engineering
Navi Mumbai, India
khairnar.vaishali3@gmail.com

***Abstract*—With increasing spread of World Wide Web, use of web service to serve various application needs has also gained popularity. While web service provides standardized method for data exchange between two software entities, making it very simplified, but at the same time maintaining security of data and assuring minimal data transfer size is very crucial. In case of security various cryptography algorithms are available but each is having some inherent limitations, in order to build a strong data security for data transmission we need to combine more than one cryptographic algorithms. Usually the dedicated Web Server hosting a web service uses SSL, but the link from the Web Server to the application or database server are generally in plain text. If the web Server falls prey to attack then all such plain text data can be hacked using a simple net-sniffing. This paper presents a method which enhances security for data by hybrid encryption, which can be used itself as end-to-end encryption or in addition to the existing SSL. This enhances security of data exchange between two clients using web service as an intermediary. The proposed solution encrypts the content block which can be only decrypted by client side, and not on the web server.**

***Index Terms*— Cryptography, Hybrid Encryption, AES, RSA, Hashing, Web Services**

1. INTRODUCTION

A web service is a standardized method (web API) for communication between two software entities over the web. [9] The web services consist of two participants, a host web application can be called as 'Service Provider', and a requesting software application can be called as 'Service Requester'. The standardization is done generally with the use of XML files for data exchange. The host application or Service Provider can be developed in any programming language; the essence of using web service is that it gives flexibility to Service Requester side to choose any programming language, not necessarily the same as the one used for developing host application.

A simple example of web service can be a stock price web service or currency exchange web service; these services facilitate clients to enquire the needed data using simple web service call.

Fig 1. Web Service Architecture Diagram

A client application can be a simple web form, which can accept input from client what to enquire. On submission client application contacts UDDI (Universal Description Discovery and Integration) provider which lookups for the service requested. UDDI creates service binding associating the message for the service request with service location, and then returns the WSDL (Web Serviced Description Language) using which application creates a SOAP request. This SOAP request is then sent over to the host application, which performs the requested operation and delivers the requested result back to the service requester. The web service fundamentally operate almost same as classic web application, apart from it is a SOAP request over HTTP. As the most of the processing is done at the host application, which processes the SOAP request and returns a response to the client, it makes it primary target for various security threats like data theft attacks, session hijacking etc.

While taking into consideration the advantages of using web services, we must also focus on two crucial aspects –Security of data being exchanged and minimizing the size of data

transfer. Therefore, it is crucial to ensure data secrecy, authenticity and integrity of the data being transferred between client and host application.

2. METHODOLOGIES

Two broad categories of cryptographic standards are 'Symmetric Key / Shared Key Algorithms' and 'Asymmetric Key / Public Key Algorithms'.

2.1 Symmetric Key Standard – AES

Symmetric Key Standards uses a same secrete key shared between sender and receiver, which will be used for both encryption as well as decryption. In Symmetric standard as the same key is used on both the ends major concerns are- key distribution and maintaining key secrecy.

Example: AES (Advanced Encryption Standard) [2] and DES (Data Encryption Standard). Not a single successful brute-force attack [5,7] on AES has been found till date, the only possible known attack against AES.

AES algorithm has fixed block size 128 bit. The key length for AES must be 128,192 or 256 bits. Number of rounds will be 10 for key size 128, 12 for key size 192, and 14 for key size 256 bits.[4] AES algorithm to encrypt 128 bit block can be summarized as below:

1. Key Expansions - Round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128 bit round key block for each round plus one more.
2. Initial Round
2.1 Add Round Key - each byte of the state is combined with a block of the round key using bitwise xor.
3. Rounds
3.1 Sub Bytes - a non- linear substitution step where each byte is replaced with another according to a lookup table.
3.2 Shift Rows - a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
3.3 Mix Columns - a mixing operation which operates on the columns of the state, combining the four bytes in each column.
3.4 Add Round Key
4. Final Round (no Mix Columns)
4.1 Sub Bytes
4.2 Shift Rows
4.3 Add Round Key

AES decryption process simply involves reversing all the steps taken in encryption. Here basically inverse functions are used at each step like Inverse Sub Bytes, Inverse Shift Rows, Inverse Mix Columns and Inverse Add Round Key.

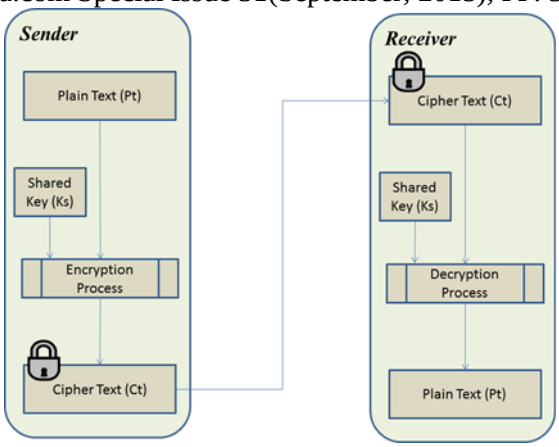


Fig. Symmetric Key Encryption AES Block Diagram

Advantages of AES:
1. Faster in both hardware and software implementations.
2. Can be used for big size data encryption.
3. Supports larger key size than its predecessors 3DES / DES.
4. Less open to attack because of large key size.

Disadvantages of AES:
1. Key exchange is major concern as the same shared key is used for both encryption and decryption
2. Prone to interpolation attack, which allows the cryptanalyst to encrypt and decrypt data for the unknown key without doing any key recovery.[8]

2.2 Asymmetric Key Standard - RSA

Asymmetric Key standards use a key pair of public key and private key. Generally, a public key is used for encryption at sender end and a private key is used for decryption at the receiver end. Also data authenticity and integrity is established with the use of digital signature. [3] The key management is an essential feature of RSA algorithm. [1] The security of the method used in RSA is based on the difficulty of factoring large numbers. [2]
Example: RSA (Rivest, Shamir, Adleman) Algorithm

RSA algorithm for encryption can be summarized as [6]:

1. Key generation:
1.1 Choose two distinct large random prime numbers p and q.

*1.2 Compute n=p*q. The number n is used as the modulus for both public and private keys*

1.3 Compute the Euler's function: z = (p −1) (q −1)

1.4 Choose an integer e, 1< e< z; such that GCD (e, z) = 1,e and z are co prime. The number e is used as a public key exponent.

*1.5 Compute d, 1< d< z such that e*d = 1 mod z. The number d is used as a private key exponent. Public key consists of*

public key exponent e and n and Private Key consists of private key exponent d and n. Public key: (e, n) and private key: (d, n).

2. Encryption - The Sender sends the original message as a plaintext (PT) and after encryption message convert to the cipher text (CT).

$$Cipher\ text\ (CT) = PT^e \bmod n$$

3. Decryption - The receiver receives the message as a cipher text (CT) and after decryption cipher text message convert to the original message (PT).

$$Plain\ text\ (PT) = CT^d \bmod n$$

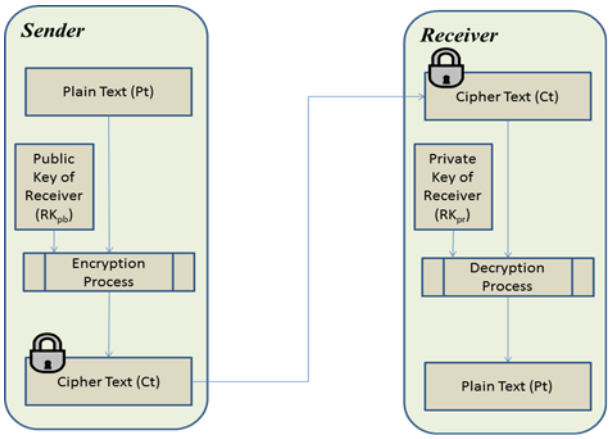


Fig. Asymmetric Key Encryption RSA Block Diagram

Advantages of RSA:
1. No key exchange problem as asymmetric key pair i.e. public key and private key pair is used.

Disadvantages of RSA:
1. More memory usage
2. Cannot be used for big size data encryption
3. Slower
4. Vulnerable to impersonation attacks

2.3 SHA-256 Hashing Algorithm

SHA-256 is a cryptographic hash function designed by the NSA (U.S. National Security Agency). SHA stands for Secure Hash Algorithm. Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity. SHA-256 hash is computed with 32 bit word.

3. PROPOSED SOLUTION

As we know, RSA cannot be used for large size data encryption, and AES is having inherent key exchange issue. While designing web service, we have to consider a security solution which will overcome the limitations of RSA as well as AES; at the same time it will also ensure to maintain data integrity. The proposed solution is a combination of symmetric and asymmetric algorithm standards, with hashing. The combination results in a hybrid algorithm which contains essence of both RSA and AES, with SHA256 hashing to ensure data integrity. The hybrid encryption and decryption between sender and receiver ensures that only legitimate data will be exchanged at the same time data cannot be tampered with.The sender side encryption terminology can be briefly summarized as below:

1. Generate an AES Session Key (K_{AES}) at runtime, use AES encryption to encrypt the plain text data (PT_{ORG}) with K_{AES}. The resulting cipher text can be called as CT_{AES}.

2. The key K_{AES} is then encrypted using RSA algorithm with receiver's Public Key (RK_{PB}). The resulting cipher key can be called as $ENCK_{AES}$.

3. The SHA256 hash of the plain text is also generated, which can be called as H_{256}.

4. Finally, an XML file is prepared consisting of CT_{AES}, $ENCK_{AES}$and H256.

The process flow at sender end can be illustrated as below:

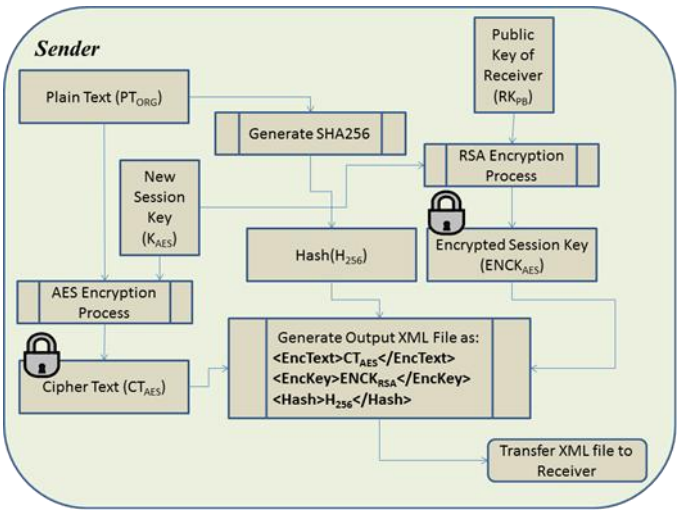


Fig. Proposed Hybrid Encryption Sender Side

The receiver side decryption terminology can be briefly summarized as below:
1. Process the received XML file and split out three components an Encrypted Cipher Text(CT_{AES}), Encrypted AES Session Key ($ENCK_{AES}$) and SHA256 has of the original plain text i.e. H_{256}.

2. The key $ENCK_{AES}$ is then decrypted using RSA algorithm with receiver's Private Key (RK_{PR}). The resulting cipher key can be called as K_{AES}.

3. The cipher text CT_{AES} is then decrypted using AES algorithm with K_{AES}. The resulting plain text can be called as PT_{DEC}.

4. Compute SHA256 hash of the PT_{DEC}, can be called as RH_{256}.

5. Match both the hashes i.e. H_{256} and RH_{256}.
If both the hashes match, it means that the decrypted data is legitimate and receiver can proceed with the decrypted data.

If the match fails, it indicates that the data is either corrupt or it is tampered, in such situation the receiver can simply discard the data.

The process flow at receiver end can be illustrated as below:

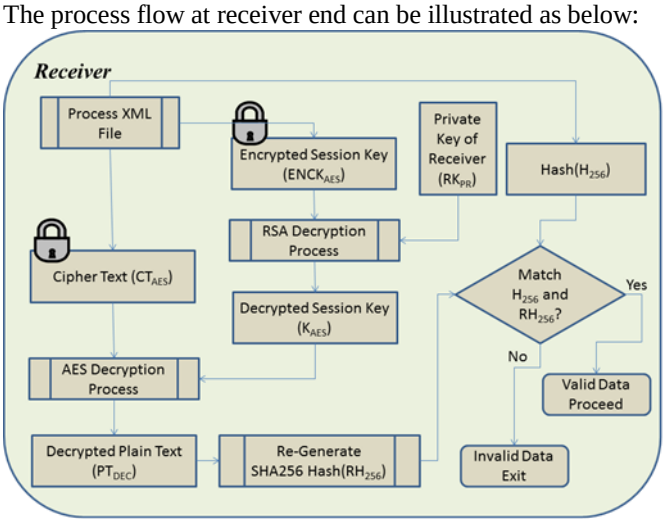


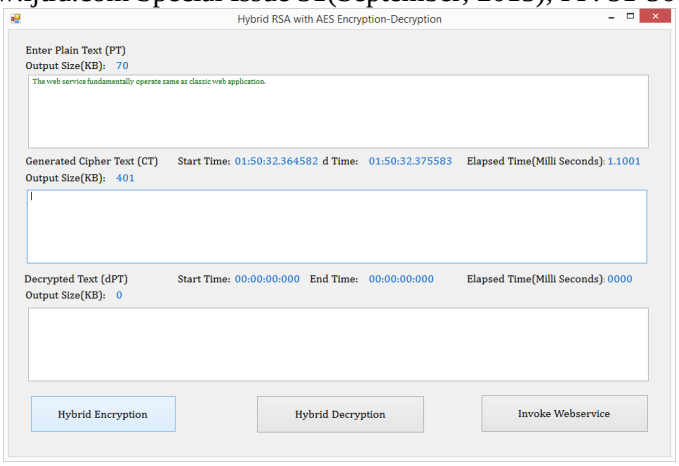
Fig. Proposed Hybrid Encryption Receiver Side

4. IMPLEMENTATION

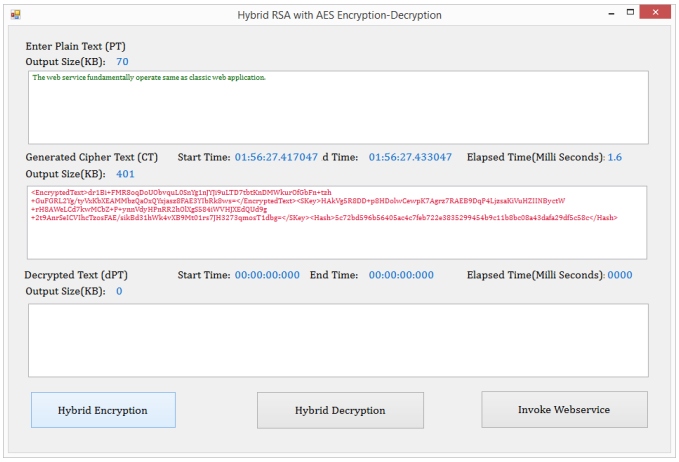
We have implemented the proposed solution, wherein two separate modules were developed – A windows application (Client side) and a Web Service (Server side). The windows application as well as the web service side we have used C# as a programming language. The web service was deployed on IIS 7.5 server. The windows application was a Windows 8 system.

The windows application program encrypts the plain text data entered by client using hybrid encryption and invokes a web service. The web service receives the data sent from windows application, and tries to decrypt the data using hybrid decryption.After data validation web service returns its response to the client program i.e. back to the windows application.The process flow can be well illustrated with following screenshots:

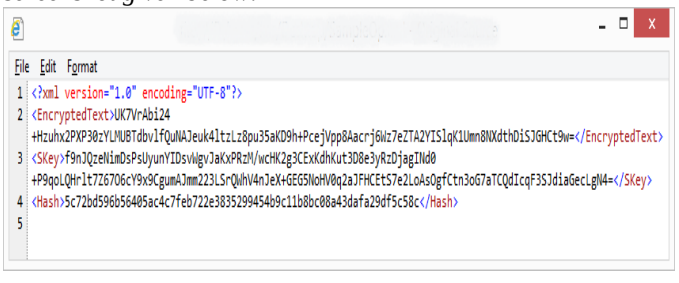
Step 1 – Client invokes a Windows Application and enters Plain Text data.



Step 2 – Client then clicks on 'Hybrid Encryption' Button to encrypt the data and prepare encrypted XML file, which can be uploaded to Web Service. This XML file contains AES Encrypted Cipher Text, RSA Encrypted Session Key and SHA256 hash.



Step 3 – Client then clicks on 'Invoke Webservice' button to invoke a web service and upload the XML data using UploadXMLEncrypted() operations defined in Web Service.XML file structure can be seen in a XML file screenshot given below:



Step 4- Web service then processes the XML file received. It first tries to decrypt the content using hybrid decryption, and then performs hash matching. After processing XML file, the web service returns the response to the client windows application as shown in screenshots below:

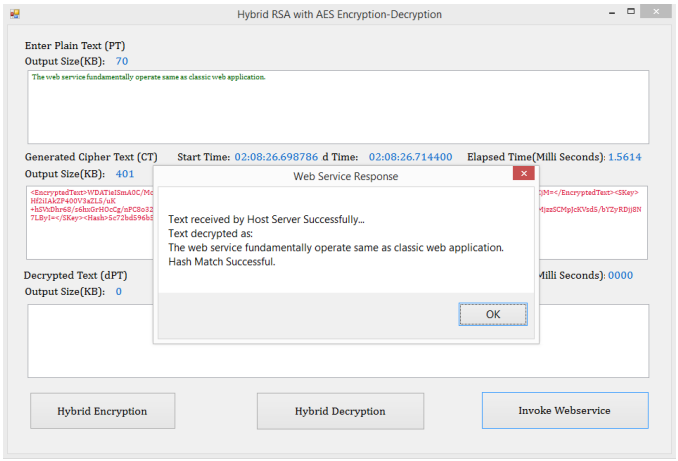


Fig. Typical success response returned from Web Service. This response indicated that the decryption was successful as well as successful hash matching.

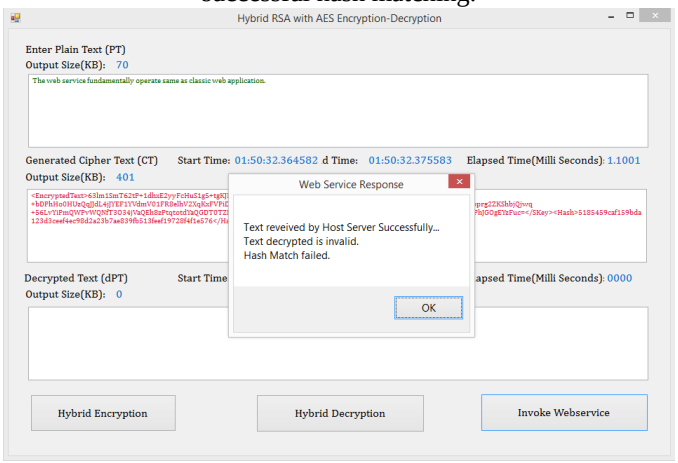


Fig. Failure response returned from Web Service, in case of hash matching fails

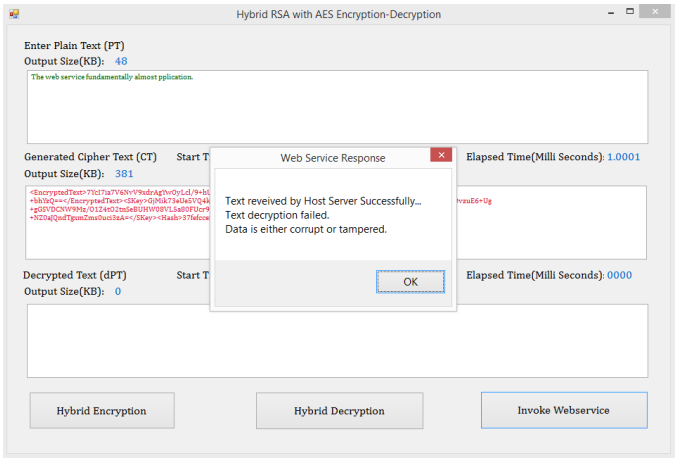


Fig. Failure response returned from Web Service, in case of decryption fails due to corrupt or tampered data

For this demonstration purpose a web service was deployed in local host:

http://localhost/aimsWSSample/Service1.svc?wsdl

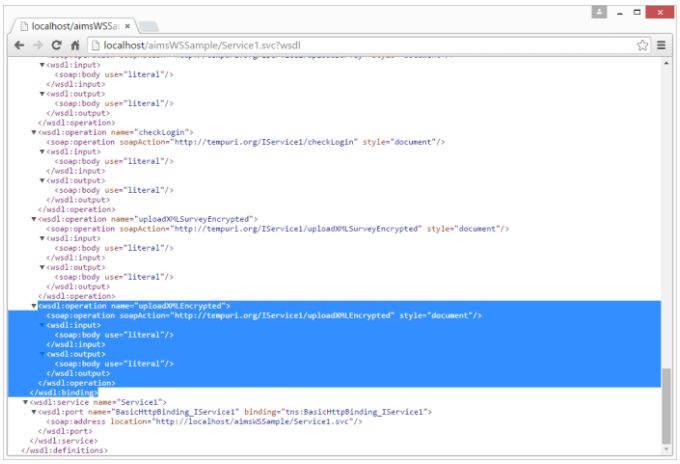


Fig. A sample WSDL showing a web service deployed on local web server

5. Conclusion

The proposed solution was successfully implemented and it is feasible to use in live scenarios. The proposed solution can be used for ensuring security and integrity while exchanging data between client applications and web service. While implementation it was observed that in normal scenarios hybrid encryption can be easily implemented and can be effectively used.

6. Future Enhancements

During implementation, we observed that the hybrid encryption output data size is comparatively higher than AES or RSA standalone implementation. Our future work will be on reducing the output data size by using some effective compression techniques.

7. Acknowledgment

With profound gratitude, we acknowledge the constant guidance and support from Prof. Vaishali Khairnar, H.O.D. Information Technology Department, Terna College of Engineering. This proposed solution was successfully implemented as result of continuous guidance from her.

References

[1] Nentawe Y. Goshwe, "Data Encryption and Decryption Using RSA Algorithm in a Network Environment", IJCSNS, VOL.13 No.7, July 2013 9

[2] Jan Mohammad Najar, Shahid Bashir Dar, "A New Design Of A Hybrid Encryption Algorithm", IJECS, ISSN:2319-7242, Volume 3 Issue 11, November, 2014

[3] https://en.wikipedia.org/wiki/RSA_(cryptosystem)

[4] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[5] William Stallings, "Network Security Essentials: Applications and Standards", Prentice Hall, 4th edition, 2011

[6] Shashikant Kuswaha, Praful B. Choudhary, Sachin Waghmare, Nilesh Patil, "Data Transmission using AES-RSA Based Hybrid Security Algorithms", IJRITCC, ISSN: 2321-8169 Volume: 3 Issue: 4, April 2015

[7] Andrew S. Tenanbaum, "Computer Networks", Fourth Edition, Pearson Education, 2003

[8] Palanisamy, V. and Jeneba Mary, A., "Hybrid Cryptography by the Implementation of RSA and AES", IJCR, Vol. 33, Issue, 4, pp.241-244, April, 2011.

[9] http://en.wikipedia.org/wiki/Web_Service