N.Veeraragavan *et al*, International Journal of Computer Science and Mobile Applications,
Vol.2 Issue. 1, January- 2014, pg. 94-100 **ISSN: 2321-8363**

PROXY RE-ENCRYPTION USING HYBRID ENCRYPTION SCHEME SECURE AGAINST CHOSEN CIPHERTEXT ATTACK

N.Veeraragavan[1], Mr.S.Rajesh[2]

[1]PG Student, [2]Assistant Professor, Department of Computer Science and Engineering,

PRIST University, Trichy District, India

([1] Veeraraghavan543@gmail.com)

Abstract

The objective is to present the unidirectional proxy re-encryption scheme with chosen-cipher text security. The goal of Proxy Re-Encryption system is to securely enable the re-encryption of cipher texts from one key to another, without relying on trusted parties. A Proxy can transform - without seeing the Plaintext - cipher text encrypted under one key into an encryption of the same plaintext under another key. Proxy Re-Encryption is a form of public-key encryption that allows a user Alice to "delegate" her decryption rights to another user Bob. In a PRE scheme, a proxy is given a piece of information that allows turning a cipher text encrypted under a given public key into an encryption of the same message under a different key.

1 INTRODUCTION

In this project we build an intranet mail application which implements SHA-512, advanced encryption standard and RSA for security over the network. SHA-512 is used for authentication of server. Advanced encryption standard is used for message encryption using secrete key. RSA is used for exchanging the secrete key. This provide confidentiality.

It provide multiple connection between several user connected in LAN. On each system it runs the client program connected with the centralized server application. Each client program in the application is capable of performing several operations over network.

- Sending mail to other user in the network
- Reading mailbox
- Forwarding mails.

The goal of our project is to provide security to the data transferred.

2 LITERATURE SURVEY

2.1 SYMMETRIC ENCRYPTION ALGORITHM

Both parties will be using the same key for encryption and decryption, called as symmetric key. This provides dual functionality. Key is shared by both sender and receiver. Symmetric keys are also called secrete key because this type of encryption relies on each user to keep the key a secret and properly protected. The security of the symmetric encryption method is completely dependent on how well users protect the key. Each pair of users who want to exchange data using symmetric key encryption must have their own set of keys.

Strengths:

Much faster than asymmetric systems.

Hard o break if using a large key size.

Weaknesses:

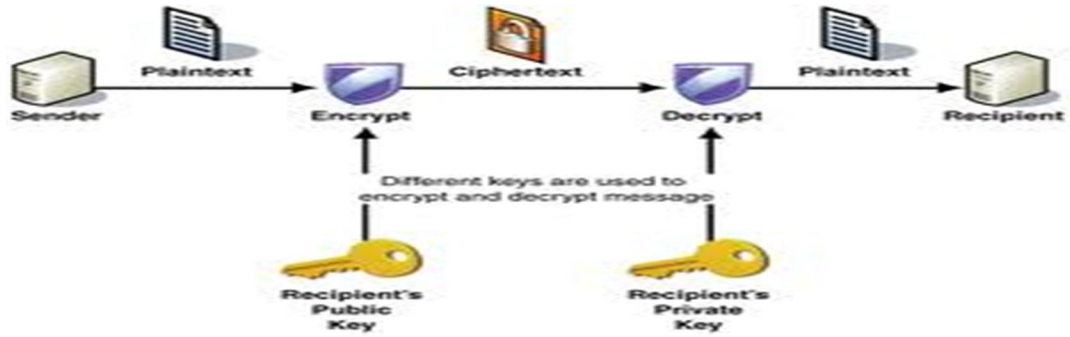
Key distribution- it requires a secure mechanism to deliver keys properly.

Scalability- each pair of users needs a unique pair of keys, so the number of keys grows exponentially.

Limited security- it can provide confidentiality, but not authenticity or no repudiation.

2.2 ASYMMETRIC ENCRYPTION ALGORITHMS

In public key system, each user has different keys, or asymmetric keys. The two different asymmetric keys are mathematically related. If a message is encrypted by one key, the other key is required to decrypt the message. In a public key system, the pair of keys is made up of one public key and one private key. The public key can be known to everyone, and the private key must only be known to the owner.



Weakness:

It does not provide more security

These algorithm woks slow.

2.3 HYBRID ENCRYPTION METHOD

A hybrid encryption scheme is one that blends the convenience of an asymmetric encryption scheme with the effectiveness of a symmetric encryption scheme.

Hybrid encryption is achieved through data transfer using unique session keys along with symmetrical encryption. Public key encryption is implemented for random symmetric key encryption. The recipient then uses the public key encryption method to decrypt the symmetric key. Once the symmetric key is recovered, it is then used to decrypt the message.

The combination of encryption methods has various advantages. One is that a connection channel is established between two users' sets of equipment. Users then have the ability to communicate through hybrid encryption. Asymmetric encryption can slow down the encryption process, but with the simultaneous use of symmetric encryption, both forms of encryption are enhanced. The result is the added security of the transmittal process along with overall improved system performance.

2.4 RSA ALGORITHM

The RSA cryptosystem is the most widely-used public key cryptography algorithm in the world. It can be used to encrypt a message without the need to exchange a secret key separately.

The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

Party A can send an encrypted message to party B without any prior exchange of secret keys. A just uses B's public key to encrypt the message and B decrypts it using the private key, which only he knows. RSA can also be used to sign a message, so A can sign a message using their private key and B can verify it using A's public key.

The RSA algorithm involves three steps: key generation, encryption and decryption.

Key generation:

RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
   - For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.
2. Compute $n = pq$.
   - n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
3. Compute $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$, where φ is Euler's totient function.
4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e. e and $\varphi(n)$ are coprime.
   - e is released as the public key exponent.
   - e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $2^{16} + 1 = 65{,}537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.[5]
5. Determine d as $d^{-1} \equiv e \pmod{\varphi(n)}$, i.e., d is the multiplicative inverse of e (modulo $\varphi(n)$).

   - This is more clearly stated as solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$
   - This is often computed using the extended Euclidean algorithm.
   - d is kept as the private key exponent.

By construction, $d \cdot e \equiv 1 \pmod{\varphi(n)}$. The public key consists of the modulus n and the public (or encryption) exponent e. Theprivate key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d.

- An alternative, used by PKCS#1, is to choose d matching $de \equiv 1 \pmod{\lambda}$ with $\lambda = \text{lcm}(p-1, q-1)$, where lcm is the least common multiple. Using λ instead of $\varphi(n)$ allows more choices for d. λ can also be defined using the Carmichael function, $\lambda(n)$.
- The ANSI X9.31 standard prescribes, IEEE 1363 describes, and PKCS#1 allows, that p and q match additional requirements: being strong primes, and being different enough that Fermat factorization fails.

Encryption:

Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send message M to Alice.

He first turns M into an integer m, such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to

$$c \equiv m^e \pmod{n}.$$

This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

Decryption

Alice can recover m from c by using her private key exponent d via computing

$$m \equiv c^d \pmod{n}.$$

Given m, she can recover the original message M by reversing the padding scheme.

(In practice, there are more efficient methods of calculating c^d using the precomputed values below.)

SECURITY OF RSA

Three possible approaches to attacking the RSA algorithm are as follows:

Brute force – use large keys

Mathematical attacks

Timing attacks

Mathematical attacks:

We can identify three approaches to attacking RSA mathematically:

- Factor n into two prime factors, this enables calculation of $\varphi(n)$ =(p-1)(q-1), which, in turn, enables determination of d=e^{-1} mod $\varphi(n)$.

- Determine $\varphi(n)$ directly, without first determining p and q.

- Determine d directly, without first determining $\varphi(n)$

Most discussions of cryptanalysis of RSA have focused on the task of factoring n into its two prime numbers. Determining $\varphi(n)$ given n is equivalent to factoring n. With presently known algorithms, determining d given e and n appears to at least as time consuming as the factoring problem. Hence, we can use factoring performance as a benchmark against which to evaluate the security of RSA.

Timing Attacks:

A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number. In this case, time for exponentiation may be used for attacking.

There are simple counter-measures against timing attacks:

- constant exponentiation time – ensure that all exponentiations take the same time, but this will degrade performance
- Random delay – better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack
- Blinding – multiply the cipher text by a random number before performing exponentiation. This process prevents the attacker from knowing what cipher text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack. RSA Data Security reports a 2 to 10% performance penalty for blinding.

Brute force attack:

A brute force attack against a cipher consists of breaking a cipher by trying all possible keys. Statistically, if the keys were originally chosen randomly, the plaintext will become available after about half of the possible keys are tried. The underlying assumption is, of course, that the cipher is known. Since A. Kerckoffs first published it, a fundamental maxim of cryptography has been that security must reside only in the key.

3 EXISTING SYSTEM

In the pre scheme, a proxy is given a piece of information that allows turning a cipher text encrypted under a given public key into an encryption of the same message under a different key. A naïve way for Alice to implement such a mechanism is to simply store her private key at the proxy when she is unavailable. When a cipher

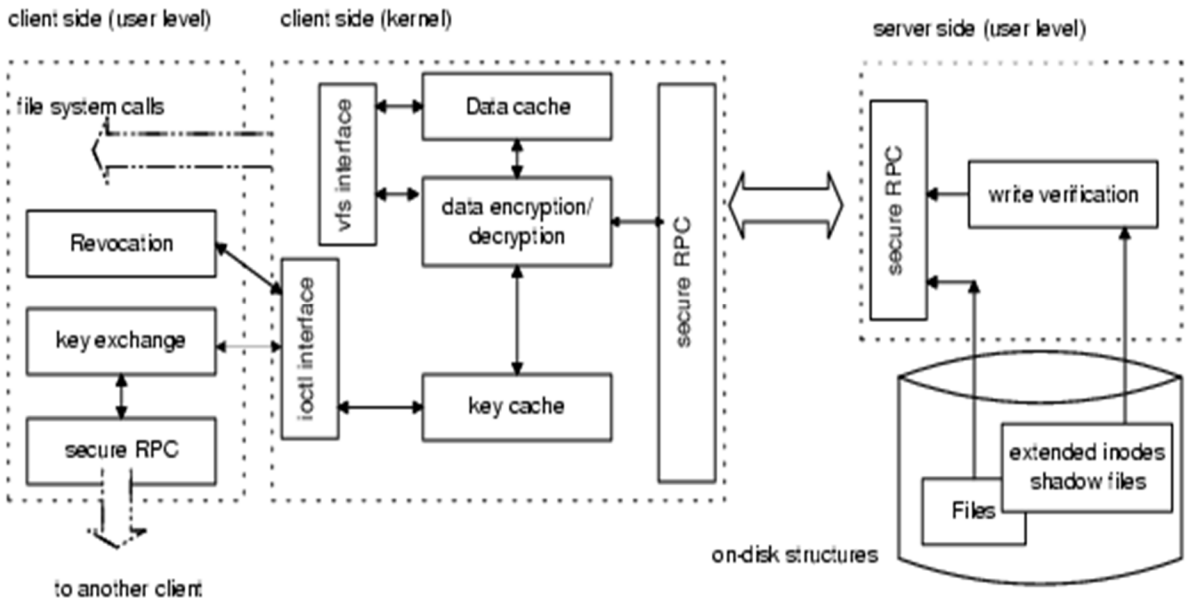
text is heading for her, the proxy decrypts it using a copy of her secrete key and re-encrypt the plain text using Bob's public key. The obvious problem with this strategy is that the proxy knows the plain text and Alice private key.

The existing system provides security against the chosen cipher text attack. The existing system was a bidirectional system.

4 PROPOSED SYSTEM

Message secrecy-proxy is given the re-encryption key. By using that proxy can re-encrypt the cipher text received for the delegator. It provides security against chosen-cipher text attacks (CCA). We are using OAEP which secures our system from being attacked using CCA. Provides solution to the key management problems. We are using symmetric algorithm (AES) for message encryption and asymmetric algorithm (RSA) for key exchange. Only intended receiver can decrypt the encrypted key.

5 SYSTEM ARCHITECTURE



6 IMPLEMENTATION

SERVER:

1. Open the Server Socket

2. Wait for the Client Request

3. Create a thread for each Client

4. Create I/O streams for communicating to the client

5. Perform communication with client

6. Close sockets

7 CONCLUSION

The project "Unidirectional Chose Cipher text Secure Proxy Re-encryption" was developed as a "intranet mail application", the system is flexible. The project provides authentication, confidentiality. The project is flexible to the network changes and adaptable to the future enhancements. As a future enhancement our project can be enhanced to support multiple Forwarding (Multiple hop) and can be used to send files.

REFERENCES

1. M. Abdalla, E. Kiltz, and G. Neven, "Generalized key delegation for hierarchical identity-based encryption," in *ESORICS'07, LNCS 4734.* New York: Springer, 2007, pp. 139–154.

2. J.-H. An, Y. Dodis, and T. Rabin, "On the security of joint signature and encryption," in *Eurocrypt'02, LNCS 2332. New York: Springer,* 2002, pp. 83–107.

3. G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy reencryption," in *CT-RSA'09, LNCS 5473. New York: Springer, 2009, pp.* 279–294.

4. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," presented at the NDSS, 2005.

6. G. Ateniese and S. Hohenberger, "Proxy re-signatures: New definitions, algorithms and applications," in *Proc. ACM Conf. Comput.Commun. Security, 2005, pp. 310–319.*

7. M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Proc. Crypto'92, 1993, pp. 390–420.*

8. http://pajhome.org.uk/crypt/rsa/implementation.html

9. http://docs.oracle.com/javase/6/docs/technotes/guides/security

10. http://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.

