ISSN 2395-1621

An Efficient Certificateless Encryption for Secure Data Sharing Over the Network Using AES-128 and AES-256

Sudhir Dhongade[#1],Shrikant Bhandare[#2],Amol Davare[#3],Rudraksh Chandel[#4]

[1]sdsudhiryash@gmail.com,
[2]shrikant.7711@gmail.com,
[3]amol.davare7344@gmail.com,
[4]rudrakshc47@gmail.com

[#1234]Deparment of Computer Engineering
JSPM's
Imperial College of Engineering & Research
Wagholi,Pune-412207

ABSTRACT

In our public network/database the practical solution to the problem is applied using our scheme. The network is employed as a secure storage as well as a key generation centre. In our system, the data owner encrypts the sensitive data using the network generated users' keys based on its access control policies and uploads the encrypted data to the network. Upon successful authorization, the network partially decrypts the encrypted data for the users. The users subsequently fully decrypt the partially decrypted data using their private keys. The confidentiality of the content and the keys is preserved with respect to the network, because the network cannot fully decrypt the information. To improve the efficiency of the encryption we propose an extension to our approach. We propose our mCL-PKE scheme, the overall network based system, evaluates its security and performance.

Keywords— **Network, certificateless cryptography, confidentiality, access control.**

ARTICLE INFO

Article History

Received:12th,September, 2015

Received in revised form :

12th September, 2015

Accepted : 15th September, 2015

Published online :

16th September 2015

I. INTRODUCTION

In order to assure confidentiality of sensitive data stored in public networks, A commonly adopted approach is to encrypt the data before uploading it to the network. Since the network does not know the keys used to encrypt the data, the confidentiality of the data from the network is assured. However a traditional public key cryptosystem requires a trusted Certificate Authority (CA) to issue digital certificates that bind users to their public keys. Because the CA has to generate its own signature on each user's public key and manage each user's certificate, the overall certificate management is very expensive and complex. To address such shortcoming, Identity-Based Public Key Cryptosystem (IBPKC) was introduced, but it suffers from the key escrow problem as the key generation server learns the private keys of all users Al-Riyami and Paterson introduced a new cryptosystem called Certificateless Public Key Cryptography (CL-PKC). Then proposed the CL-PRE (Certificateless Proxy Re-Encryption) scheme for secure data sharing in public network environments. Although their scheme implement me is based on CL-PKC to solve the key escrow problem and certificate management, it relies on pairing operations. Despite recent advances in implementation techniques, the computational costs required for pairing are still considerably high we address the shortcomings of such previous approaches and propose a novel mediated Certificateless Public Key Encryption (mCL-PKE) scheme that does not utilize pairing operations. Since most CL-PKC schemes are based on bilinear pairings, they are computationally expensive. Our scheme reduces the computational overhead by using a pairing-free approach. Further, the computation costs for decryption at the users are reduced as a semi-trusted security mediator partially decrypts the encrypted data before the users decrypt. The security mediator acts as a policy enforcement point as well and supports instantaneous revocation of compromised or malicious users. In this, we show that our scheme is much more efficient than the pairing based scheme proposed.

Each user only needs to maintain its public/ private key pair, in our approach private keys of the users are not required to be changed. Based on our mCL-PKE scheme, we propose a novel approach to assure the confidentiality of data stored in public networks while enforcing access control requirements.

There are five entities in our system: the data owner, users, the Security Mediator (SEM), the Key Generation Center (KGC), and the storage service. The SEM, KGC, and the storage service are semi-trusted and reside in a public network. Although they are not trusted for the confidentiality of the data and the keys, they are trusted for executing the protocols correctly. According to the access

control policy, the data owner encrypts a symmetric data encryption key using mCL-PKE scheme and encrypts the data items using symmetric encryption algorithm. Then, data owner uploads encrypted data items and the encrypted data encryption key to the network. Notice that a major advantage of our approach compared to conventional approaches is that the KGC, which is the entity in charge of generating the keys, resides in a public network. Thus it simplifies a task of key management for organizations. In a conventional CL-PKE scheme, user's complete private key consists of a secret value chosen by the user and a partial private key generated by the KGC. Unlike the CLPKE scheme, the partial private key is securely given to the SEM, and the user keeps only the secret value as its own private key in the mCL-PKE scheme. So, each user's access request goes through the SEM which checks whether the user is revoked before it partially decrypts the encrypted data using the partial private key. It does not suffer from the key escrow problem, because the user's own private key is not revealed to any party. It should be noted that neither the KGC nor the SEM can decrypt the encrypted data for specific users. Moreover, since each access request is mediated through the SEM, our approach supports immediate revocation of compromised users. In our extension, the cloud simply acts as storage and does not perform any transformation. Instead, the user is able to decrypt using its own private key and an intermediate key issued by the data owner.

II. RELATED WORK

A.) AES-128

The basic unit of AES algorithm process is a byte and this algorithm is based on Substitution Permutation network it means it has series of linked mathematical operations. AES consist of two dimensional array called as state. The AES algorithm is basically used in A TM machines for security of transactions .It is also used in Windows Vista fault analysis program software to provide configuration file security. AES mainly proposed in many platforms of languages i.e. Matlab, C and Java. The AES algorithm has 4 phases that execute the process in sequential manner. The encryption process is achieved by processing plain text and key for initial and 9 rounds, Same decryption is takes place but in reverse manner. A 4x4 state is formed in each round and particular length data is introduced init for encryption process.

B.) AES -256

AES-256 is a symmetrical encryption algorithm that has become ubiquitous, due to the acceptance of the algorithm by the U.S. and Canadian governments as standards for encrypting transited data and data at rest. Because of the length of the key (256 bits) and the number of hashes, it takes a murderously long time for a malware hacker to perform a dictionary attack. Inferences of a stream or stored data won't likely happen in your lifetime, or in the next hundred lifetimes. Plus, it's straightforward for developers to use. Much of the AES256 plumbing has already been done for you in the form of numerous libraries that call this encryption methodology in a way that's easy to implement. Just don't lose the keys. If you lose the keys, you lose the data at rest or the conversation *completely and irrevocably*. It's that simple. Advanced Encryption System 256 uses a 256-bit encryption key, which is the same key for both directions of data encrypted. That is, *to* and *from* uses the same key and at-rest *written* and *read* uses the same key. Hashing data makes it iteratively more difficult to be revealed. More hashing is better. Hashing data 14 times, as done with AES256, used to take up a lot of CPU time, but as systems become faster and with more cores, it's become transparent in use for at least the client side of notebooks, desktops, tablets, and even smartphones. Servers running multiple instances of the algorithm can be slowed, depending on the duty cycle of transactions and their concurrency. It's believed you cannot crack the encryption behind *correctly* proposed AES256 unless you employ binoculars, have a really good psychic, or you're measuring keystroke bounces. The algorithm's implementations and related processes have been cracked, now and again, due to mistakes someone made when using the algorithms incorrectly or by making non-random key seeds. As **A race to processing power** attacks the algorithm, it's still impossible to crack within a lifetime without extraordinary (and highly unlikely) processing power. Only the NSA knows in certainty, however, if it's rationally uncrackable.

III. EXISTING SYSTEM

The Existing System CL-PRE (Certificateless Proxy Re-Encryption) scheme for secure data sharing in public network/database. Although their scheme is based on CL-PKC to solve the key escrow problem and certificate management, it relies on pairing operations. Despite recent advances in implementation techniques, the computational costs required for pairing are still considerably high compared to the costs of standard operations such as modular exponentiation in finite fields.

Disadvantages of existing system:

- In addition to the key escrow problem, ABE has the revocation problem as the private keys given to existing users should be updated whenever a user is revoked.
- Moreover, their scheme only achieves Chosen Plaintext Attack (CPA) security. As pointed out, CPA security is often not sufficient to guarantee security in general protocol settings. For example, CPA is not sufficient for many applications such as encrypted email forwarding and secure data sharing that require security against Chosen Cipher text Attack.

IV. PROPOSED SYSTEM

It is important to notice that if one directly applies our basic mCL-PKE scheme to network and if many users are authorized to access the same data, the encryption costs at the data owner can become quite high. In such case, the data owner has to encrypt the same data encryption key multiple times, once for each user, using the users' public keys. To address this shortcoming, we introduce an extension of the

basic mCL-PKE scheme. Our extended mCL-PKE scheme requires the data owner to encrypt the data encryption key only once and to provide some additional information on the network so that authorized users can decrypt the content using their private keys. Our proposed system gives a high-level view of the extension. The idea is similar to Proxy Re-Encryption (PRE) by which the data encryption key is encrypted using the data owner's public key and later can be decrypted by different private keys after some transformation by the database which acts as the proxy. However, in our extension, the database simply acts as storage and does not perform any transformation. Instead, the user is able to decrypt using its own private key and an intermediate key issued by the data owner.

Advantages of proposed system:

- We present the formal security model and provide the security proof. Since our mCL-PKE scheme does not depend on the pairing-based operation, it reduces the computational overhead.
- Unlike conventional approaches, the KGC only needs to be semi-trusted and can reside in the public network, because our mCL-PKE scheme does not suffer from the key escrow problem.

V. MATHEMATICAL MODEL

Module 1:- User Key Generated and Encrypted
Let S1 be a set of parameters for generating key and cypher text
S1= {key}
Generating key:- R=((N-NP) S/L)/N=S/L

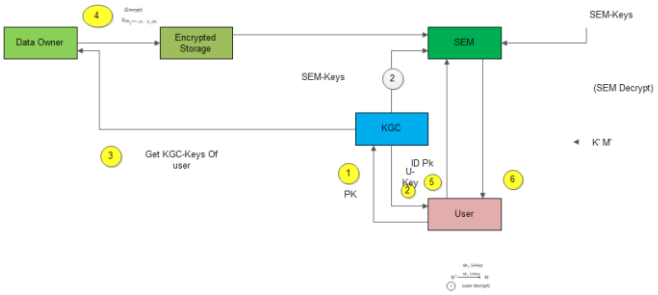
where, R is Binary data rate, N is Size of key, NP is size of key which carries the parameters, S is Small positive integer and L is size of binary data in key data.

Where,
File_Size = Actual size of key
key_Type = Type of key

Condition/Parameter	Operation/Function
If key_Type==Encrypted	f1:Proceed()
Else..	Discard Operation

If key is encrypted then proceed Else discard operation.

VI. SYSTEM ARCHITECTURE



Each user first generates its own private and public key pair, called PK and PubK, using the SetPrivateKey and SetPublicKey operations respectively using our mCL-PKEscheme. The user then sends its public keys and its identity (ID) to the KGC in the network. The KGC generates two keys for the user. One key, referred to as *SEM-key*, is stored at the SEM in the network. The other key, referred to as *User-key*, is given to the user. The data owner obtains the KGC-keys of users from the KGC in the network. The data owner then encrypts each data item for which the same access control policy applies using a key *K* and then the data owner encrypts *K* using the KGC-keys of users. The encrypted data is uploaded to the network. When a user wants to read some data, it sends a request to the SEM to obtain the partially decrypted data. The SEM first checks if the user is in the access control list and if yes then user's KGC-key encrypted content is available in the network storage. If the verification is successful, the SEM retrieves the encrypted content from the network and partially decrypts the content using the SEM-key for the user. The partial decryption at the SEM reduces the load on users. The user uses its SK and U-key to fully decrypt the data.

VII. CONLUSION

In this paper we have proposed the aes-128-256 scheme without pairing operations and provided its formal security. Our aes-128-256 solves the key escrow problem and revocation problem. Using the AES-128-256 scheme as a key building block, we proposed an improved approach to securely share sensitive data in our database. Our approach supports immediate revocation and assures the confidentiality of the data stored in an untrusted database while enforcing the access control policies of the data owner. Our experimental results show the efficiency of basic AES -128-256 scheme and improved approach for the database. Further, for multiple users satisfying the same access control policies, our improved approach performs only a single encryption of each data item and reduces the overall overhead at the data owner.

VII. REFRENCES

1) Abha Sachdev, " Enhancing Cloud Computing Security using AES Algorithm" International Journal of Computer Applications (0975 – 8887) Volume 67– No.9, April 2013
2) T. St Denis and S. Johnson, "Advanced Encryption Standard," in Cryptography for Developers, ed Burlington: Syngress, 2006, pp. 139- 202.
3) S.Heron,"Advanced Encryption Standard(AES),"Network Security,Vol.2009,pp.8-12-2009
4) F.I.P.S.197,"Advanced Encryption Standard (AES) (FIPS PUB 197)," ed: National Institute of Standards and Technology, November 2001, pp. 1-51.
5) S. Al-Riyami and K. Paterson,"Certificateless public key cryptography," in *Proc. ASIACRYPT 2003*, C.-S. Laih, Ed. Berlin, Germany: Springer, LNCS 2894, pp. 452–473.
6)] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Proc. Crypto '98*, H. Krawczyk Ed. Springer-Verlag, LNCS 1462
7) Suchita Tayde," File Encryption, Decryption Using AES Algorithm in Android Phone," Volume 5, Issue 5, May 2015

