2FA (Factor Authentication) for Cloud Storage System

Manoj K
VIII Semester,
Dept. of Computer Science and Engineering
Siddaganga Institute of Technology
Tumkur, India.

Jaishankar K V
VIII Semester,
Dept. of Computer Science and Engineering
Siddaganga Institute of Technology
Tumkur, India.

Venugopal S A
VIII Semester,
Dept. of Computer Science and Engineering
Siddaganga Institute of Technology
Tumkur, India.

Chandraprabha K S
Assistant Professor,
Dept. of Computer Science and Engineering
Siddaganga Institute of Technology
Tumkur, India.

***Abstract*— Storing data in a cloud is very much important to assist user in many ways. To ensure the security to the stored data is much more crucial. So, here we introduce a two-way authentication for data protection with a revocability of the device used for cloud storage system. Our system allows the sender to send his information in cloud with dual encrypted format. The sender has to aware of the receiver's information (such as his email). The receiver needs to possess two things in order to decrypt the cipher text. The first thing is his/her secret key is sent to his/her email-id. The second thing is a unique personal security device (such as USB) which will be connected to the computer. It is impossible to decrypt the cipher text without either piece. More importantly, revocability factor for the device is provided if the device is lost, it cannot be possible to decrypt any cipher text. This can be done by the proxy server which will immediately execute DES algorithms to change the existing plain text to be un-decrypt-able form. This process is not known to sender because of unaware of the secret key and unique device. Furthermore, the cloud server cannot decrypt any cipher text at any time. The security and efficiency analysis shows that our system is not only secure but also practical.**

Keywords*— Secret key, double encryption, unique security device, revocability, unique security device id

I. INTRODUCTION

Cloud storage [5], [4], [6], [5], [1] is a model of networked storage system where data is stored in pools of storage which are generally hosted by third parties. There are many advantages to use cloud storage. The most important is data accessibility. Data stored in the cloud can be accessed at any time from any place as long as there is network access. Storage maintenance tasks, such as purchasing additional storage capacity, can be offloaded to the responsibility of a service provider. Another advantage of cloud storage is data sharing between users.

In a normal asymmetric encryption, there is a single secret key corresponding to a public key or an identity. The decryption of cipher text only requires this key. The key is usually stored inside the proxy trusted server, and may be protected by a password. The security protection is sufficient if the computer/server is isolated from an opening network. Unfortunately, this is not what happens in the real life. When being connected with the world through the Internet, the computer/server may suffer from a potential risk that hackers may invade into it, to compromise the secret key without letting the key owner know. In the physical security aspect, the computer storing a user decryption key may be used by another user when the original computer user (i.e. the key owner) is away. In an enterprise or college, the sharing usage of computers is also common. For example, in a college, a public computer in a copier room will be shared with all students staying at the same floor. In these cases, the secret key can be compromised by some attackers who can access the victim's personal data stored in the cloud system. Therefore, there exists a need to enhance the security protection. The purpose of using two factors is to enhance the security protection for the access control. As cloud computing becomes more mature and there will be more applications and storage services provided by the cloud, it is easy to foresee that the security for data protection in the cloud should be further enhanced [6], [4], [2], [3]. They will become more sensitive and important. Actually, we have found that the mechanism of dual-factor encryption, which is one of the encryption mechanism for data protection1, has been spread into some real-world applications, for example, full disk encryption with Ubuntu system, AT&T two factor encryption for Smartphones2, electronic vaulting and druva - cloud-based data encryption3. However, these system suffer from a potential risk about mechanism device revocability that may limit their practicability functionality. A flexible and scalable two factor encryption mechanism is really desirable in the era of cloud computing. That motivates our work.

Here we have some naive approaches for enhancement of security protection and explain why they are not the best candidate to achieve the goal of flexibility.

1) Double encryption: A security device (with an additional public key or serial number) is still required. The encryption process is executed twice. First encrypt the plaintext with the

public key or identity of the user. Then encrypt it again corresponding to the serial number of the unique device. For the decryption stage, the unique device first decrypts once. The partially decrypted cipher text is then passed to the computer which uses the user secret key to further decrypt it. Without either part (user secret key or security device) one cannot decrypt the cipher text. It seems that this naive approach can achieve our goal. However, there exist many practical issues that it cannot solve. For example,

- If the user has lost his security device, then his/her corresponding cipher text in the cloud cannot be decrypted forever! That is, the approach cannot support security device update/ revocability.
- The sender has to be aware of serial number of the security device, in additional to the user's identity. That makes the encryption process more complicated. In the case of identity-based encryption, the concept of "identity-based" has been totally lost as the sender needs to know not only the identity but another serial number! 2)divide the secret key into two parts: Another naïve way to think of is to just divide the secret key into two parts.

2) Split the secret key into two parts: Another naïve way to think of is to simply split the secret key into two parts. The first part is stored in the computer while the second part is embedded into a security device. Similar to the above approach, without either part one cannot decrypt the cipher text.

Again it seems that this approach can achieve our goal. However, note that the security of a normal encryption scheme cannot be guaranteed if part of the secret key has been exposed. The security is only guaranteed if the secret key has not exposed to the vulnerability. In other words, if we just divide the secret key into two parts, the effect is with either part may have non negligible chance to decrypt (or at least to know some information about the plaintext).

II. RELATED WORKS

There exists another cryptographic primitive called "leakage-resilient encryption" [1], [6], [3]. The security of the scheme is still guaranteed if the leakage of the secret key is up to certain bits such that the knowledge of these bits does not help to recover the whole secret key. However, though using leakage resilient primitive can safeguard the leakage of certain bits, there exists another practical limitation. Suppose we put part of the secret key into the security device. Unfortunately the device is stolen. The user needs to obtain a replacement device so that he can continue to decrypt his corresponding secret key. The trivial way is to copy the same bits (as in the stolen device) to the new device by the private key generator (PKG). This approach can be easily achieved. Nevertheless, there exists security risk. If the adversary (who has stolen the security device) can also break into the computer where the other part of secret key is stored, then it can decrypt all cipher text corresponding to the victim user. The most secure way is to cease the validity of the stolen security device.

III. ASSUMPTIONS

In this paper, we consider the following threats:

1) Type-I: Decrypt without security device: The adversary tries to decrypt the cipher text without the security device, or using a revoked security device, or using another security device belonging to others. It can have its own secret key.

2) Type-II: Decrypt without secret key: The adversary tries to decrypt the cipher text without any secret key. It can have its own security device. Note that the above threat model has already captured the semi-trust behavior of the cloud server.

IV. PROBLEM FORMULATION

Construction Roadmap: We leverage two different encryption technologies: one is IBE and the other is traditional Public Key Encryption (PKE). We first allow a user to generate a first level cipher text under a receiver's identity. The first-level cipher text will be further transformed into a second level cipher text corresponding to a security device. The resulting cipher text can be decrypted by a valid receiver with secret key and security device. Here, one might doubt that our construction is a trivial and straightforward combination of two different encryptions. Unfortunately, this is not true due to the fact that we need to further support security device revocability. A trivial combination of IBE and PKE cannot achieve our goal. To support revocability, we perform re-encryption mechanism such that the entire secret key for an old security device can be flashed for a new device if the old device is revoked. Meanwhile, we need to generate a special key for the above cipher text conversion. We also guarantee that the cloud server cannot achieve any knowledge of message by accessing the special key, the old cipher text and the updated cipher text. We further use hash-signature method to "sign" cipher text such that once a component of cipher text is tempered by adversary, the cloud and cipher text receiver can tell. From the above presentations, we can see that our two factor protection system with security device revocability cannot be obtained by trivially combining an IBE with a PKE. We present the system description as shown in the fig 1.

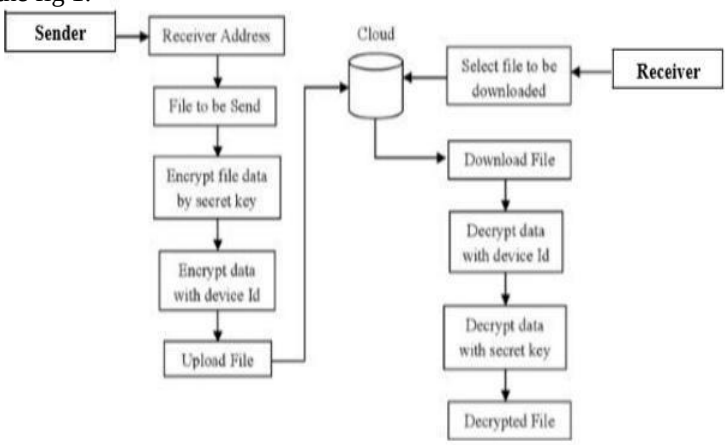

Fig. 1: Structure chart of two factor data security mechanism.

The Design of the proposed method contains several phases.

1) Setup Phase: the setup phase generates all public parameters and master secret key used throughout the execution of system. The public parameters are shared with all parties participating into the system (including data sender/receiver, cloud server and a PKG), while the master secret key is given to the PKG.

2) Key and Device Issued Phase: A SDI (Security Device ID) and a PKG will respectively generate a security device and a secret key for a registered user ID_i in secure channel such that the user can combine the security device with the secret key to recover message from its encrypted format.

3) First-Level Cipher text Generation Phase: a data sender encrypts a data under the identity of a data receiver, and further sends the encrypted data to the cloud server

4) Second-Level cipher text Phase: after receiving the first-level cipher text of a data from the data sender, the cloud server generates the second-level cipher-text. Knowing public parameters, a first level encryption for the user, and the information (ID_i, tpk_i) stored in List, the cloud server encrypts $C_1 = (c_1, c_2, c_3, c_4)$ to a second-level cipher text.

5) Device Updated Phase: Once a device of a user needs to be updated due to some incidences (e.g. it is either lost or stolen), the user first reports the issue to the SDI. The SDI then issues a new device for the user.

6) Cipher text Updated Phase: The SDI notifies the cloud server to update the cipher text of the user by sending a special piece of information.

a) The SDI first sends a piece of information to the cloud server so as to inform the cloud to execute the cipher text updated process.

b) After receiving the information, the cloud server updates the cipher text C2.

7) Data Recovery Phase. A data receiver uses a decryption key and a device to recover the data.

IV. IMPLEMENTATION

The implementation work flows contains sender module and receiver module.

Sender Module:

This module is responsible for Registration (Setup Phase), Secret Key Generation, KDC Request and then File Uploading to the Amazon Cloud Server.

• Registration :

– Functionality : Sender details are registered
– Input : EmailId, Phone Number, UserName, Password
– Output : Upon registering, successful message is displayed and secret key is sent to the Receiver EmailId.

If the user is a new user, then secret key is also generated.

• KDC Device Request :

– Functionality : USB Security device is registered using key distribution center.
– Input : USB Security device
– Output : Flashing of USB Security device

Using SHA Algorithm(Secured Hash Algorithm), Device identity is generated.

• File Upload :

– Functionality : Sender uploads the file to the Amazon Cloud Server by encrypting the file content by two time.
– Input : Filename
– Output : File uploaded to Amazon Cloud Server bucket.

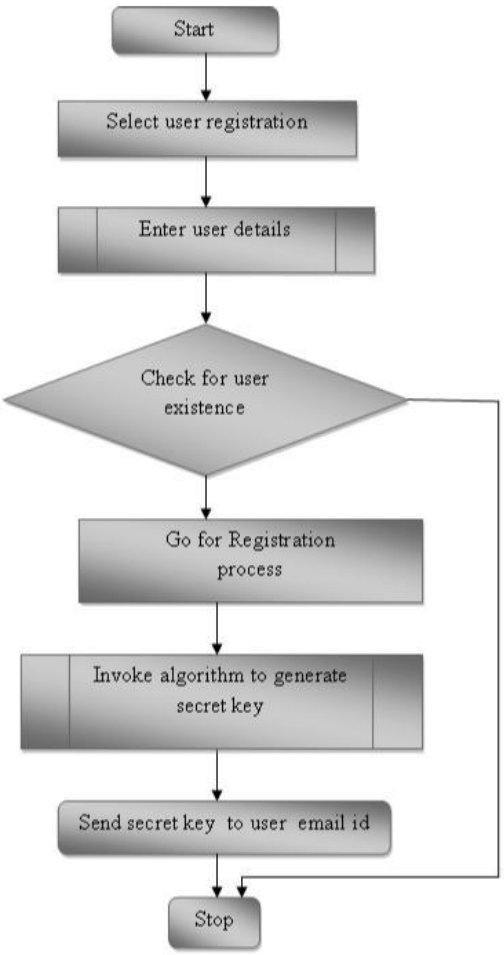

Fig 2. Flow chart for user Registration with a Secret key generation.

Receiver Module:

This module is responsible for File Download and File Decryption.

• File Download :

– Functionality : File is downloaded from Amazon Cloud Server to local machine and then decrypted by two time.
– Input : Filename
– Output : File is downloaded to local machine

Before decrypting the file, USB device is checked for the authenticity and similarly secret key is also checked for whether it is valid or not. Afterwards, File is decrypted two times.

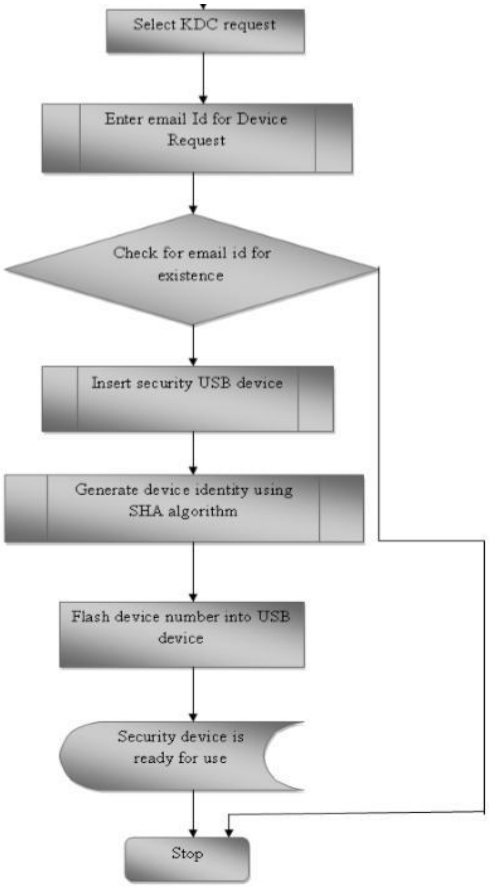

Fig.3 Flow chart for KDC device request

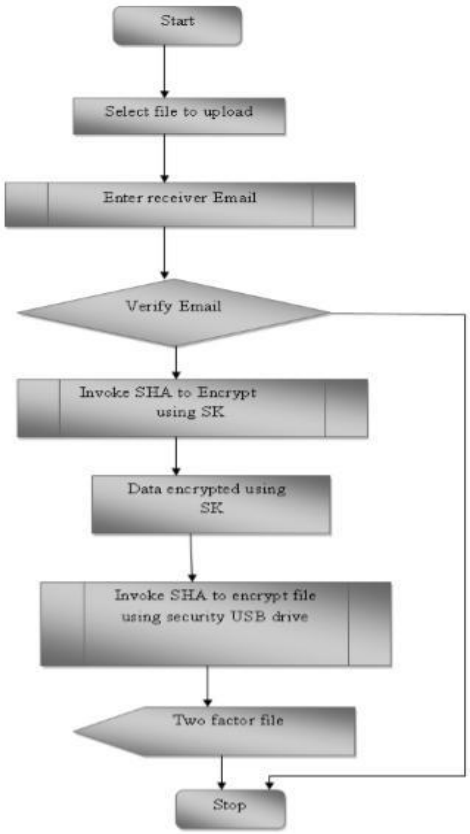

Fig 4. Flow chart for File Upload

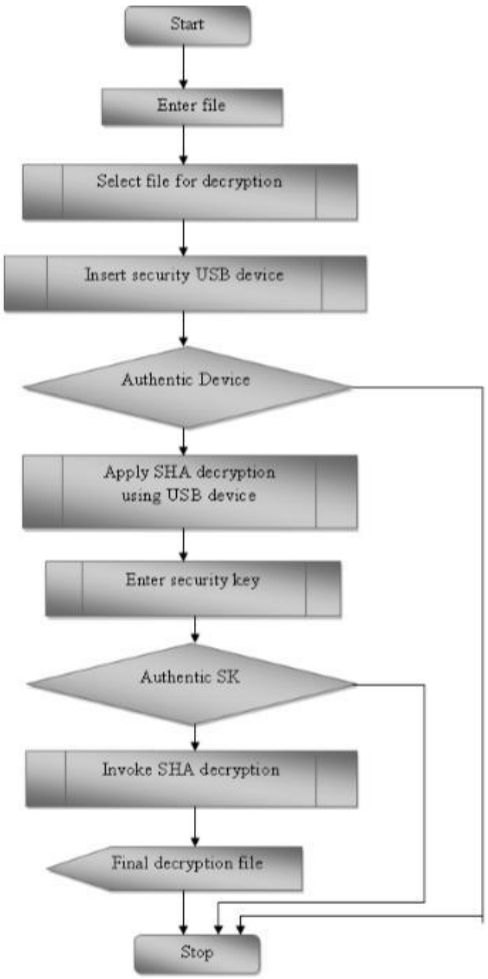

Fig 5. Flow chart for File Download.

V. CONCLUSION

High valuable sensitive data can be protected by using this Two-factor authentication system. Company's commercial secret or Personal genome information etc can be protected very well using this system. In this system, sender needs to know only the identity of the receiver like EmailId, Phone number etc. Using this identity, sender encrypts the data twice and uploads the data to cloud server. The receiver then uses USB security device issued by security device issuer and secret key to decrypt the downloaded data. The project also supported the revocability of the device. If the USB security device is lost, then it can be replaced with the new USB security device. The computation time taken to convert the plaintext to ciphertext is also very less compared to other system. Around 0.00548 seconds was taken to convert the plaintext to ciphertext, using 64-bit length of secret key size.

REFERENCES

[1] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In CRYPTO '01, volume 2139 of LNCS, pages 213– 229. Springer, 2001.

[2] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, ACM Conference on Computer andCom-munications Security, pages 185–194. ACM, 2007.

[3] S. S. M. Chow, C. Boyd, and J. M. G. Nieto. Security-mediated certificateless cryptogra-phy. In Public Key Cryptography, volume 3958 of Lecture Notes in Computer Science, pages 508–524. Springer, 2006.

[4] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng. Key-aggregate cryp-tosystem for scalable data sharing in cloud storage. IEEE Trans. Parallel Distrib. Syst., 25(2):468–477, 2014.

[5] C. Gentry. Certificate-based encryption and the certificate revo- cation problem. In EU-ROCRYPT, volume 2656 of Lecture Notes in Computer Science, pages 272–293. Springer, 2003.

[6] J. K. Liu, M. H. Au, and W. Susilo. Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model: extended abstract. In ASIACCS, pages 273–283. ACM, 2007.