A CAPABLE ACCESS CONTROL SYSTEM FOR PROTECTED CLOUD STORAGE ANTI ATTACKING DATA

Mr. G Naga Kumar Kakarla[1], Spurthi K [2], Dr.Kishore Kumar K[3]

[1]**Associate Professor, Computer Science and Engineering Gokaraju Lailavathi Womens Engineering College Hyderabad, 500049**

[2]**Assistant Professor, Dept of CSE, Siddhartha Institute of Technology & Sciences, Narapally, Hyderabad, 500088, India.**

[3]**Associate Professor Dept of CSE, Siddhartha Institute of Technology & Sciences, Narapally, Hyderabad, 500088, India.**

ABSTRAT

Given that it can significantly reduce the cost of hardware and software resources, one of the most popular ideas in the IT sector is cloud computing, which refers to computing infrastructure. Thanks to this simplicity of use, companies can now successfully capture data among their employees. At first, the simplest solution seemed to simply be to store shared, unencrypted copies of archived data in the cloud while keeping it securely accessible. The basis for this is the false assumption that clouds maintained by a third party can be trusted. Complete confidence that all information will remain confidential. Therefore, encryption is necessary and shared access control must be used to store data in cipher text. However, in reality, some of these employees may be rude and unwilling to follow basic engagement guidelines. Unfortunately, issues caused by weak data providers are not currently being addressed. Information stored in cloud storage can only be decrypted by authorized recipients thanks to literature describing the current form of protection. Although malicious data editors write data according to regulations, encrypted texts can be decrypted without their knowledge by unauthorized users who have the right keys or simply by anyone who should not have the keys. The use of malicious data editors has a negative impact because it may put a company's intellectual property at risk. To this day, the goal of the study is to determine how to provide a reasonable solution to the problem when there are conflicting data producers in the system.

Keywords: Secure cloud storage, access control, sanitizable, malicious data publishers

I INTRODUCTION

The emergence of cloud storage technology has greatly impacted business operations and the adoption of cloud technology has been one of the biggest changes in the digital era. Cloud storage provides low-cost solutions, which would be ideal for businesses, such as small and medium-sized enterprises (SMEs). Having cloud storage allows businesses to easily share their data among their employees. This data is supposed to be used only by employees of those companies, because it may be linked to their intellectual property. At first glance, simply storing data as plain text in the cloud and protecting it with proper access control would be a sufficient solution. This is based on the assumption that the cloud is completely trustworthy and will not leak that data, which is impractical since the cloud is owned by a third party. Therefore, it is necessary to use encryption mechanism and store data as ciphertext in the cloud to prevent data leakage. An existing body of work in the literature leverages the idea of attribute-based encryption (ABE) [1], [2], [3] to enable this unauthorized prevention by protecting the data with an appropriate access policy. Anyone with a valid decryption key that complies with the access policy will be able to successfully decrypt the data. This means that the data will be stored as ciphertext instead of plaintext in the cloud. This type of protection takes into account data privacy only when the data publishers are honest and follow the encryption algorithm. Unfortunately, in practice, some of these employees may be malicious and attempt to intentionally leak the contents of that data to unauthorized recipients, such as competing businesses. These malicious employees may want to publish some sensitive content and store it in the cloud, but they also allow other unauthorized users to retrieve it, thus constituting a malicious data spreader. Unfortunately, the ABE-based approach is not robust due to malicious data editors, which maliciously encrypt data. Here, a malicious data editor creates encrypted data according to the specified policy, but the ciphertexts can be decrypted by unauthorized users without valid keys. In practice, malicious data spreaders refer to company employees infected with viruses or computers intended to leak sensitive internal information. For example, a malicious data editor might want to leak new product designs or trade secrets that only certain people are supposed to have access to.

The impact of malicious data sharing by publishers is detrimental. In the above setup, malicious data editors will create ciphertexts containing copyrighted material that appear to comply with the required security access policy set by the organization. However, an illegally encrypted file can be decrypted by anyone without a valid decryption key. Therefore, our main goal is to achieve data privacy when data publishers are malicious and do not follow the encryption algorithm accordingly. Our goal is to propose a very practical idea, called a sanitizable access control system, or simply SACS, which is designed to make cloud storage resilient against malicious data editors. SACS enables flexible access control for both publishers and data recipients. Similar to ABE, SACS allows any valid recipient equipped with private keys that comply with the access policy to decrypt the ciphertext. However, SACS is equipped with a sanitization capability, which prevents malicious data editors from creating ciphertexts that can be decrypted without any valid private key. Although malicious data miners can maliciously create ciphertexts that anyone can decrypt, a sanitizer will turn these ciphertexts into new ciphertexts that can only be decrypted by holders of valid private keys. We present our architecture and plan to realize the above concept of building SACS. In addition, we also offer the SACS application. organized. The rest of this document is organized as follows.

II EXISTING SYSTEM

Currently, any file or document that a customer stores in a cloud computing environment is completely vulnerable to hacking, giving a hacker access to the entire contents of the file. The five types of entities that make up the system model are cloud, user, sanitizer, private key generator (PKG), and external auditor. Users have the ability to share their data with others by uploading it to the cloud and using the storage space provided by the service. In the cloud, the user can maintain large amounts of data. The user works for a company that requires storing a lot of data in the cloud. Data blocks in the file that represent sensitive information (personally identifiable information) and the sanitizer must verify signatories' signatures in order to access sensitive organizational data.

Cloud storage users have the option to save their content to the cloud remotely and make it available to others. A cloud storage system composed of many storage servers provides long-term online storage services. When data is stored on third-party cloud computing infrastructure, there is a significant risk to data privacy. Allowing a third party auditor (TPA) to confirm, on behalf of the customer, the accuracy of dynamic data stored in the cloud. Previous attempts to maintain remote data integrity often lacked provision for dynamic data processes or the ability for public auditing.

III LITERATURE SURVEY

In this section, we review some closely related literature. Access Control: Access control is able to ensure the security of data in cloud storage systems. This has attracted a lot of attention from academia and industry. IBM developed the model based on systematic capabilities and approaches to improve access control in cloud services. Cryptographic primitives have been proposed to enable access control to encrypted storage, such as stream encryption, proxy re-encryption, role-based encryption, and attribute-based encryption. For reasons of security, scalability, and flexibility, ABE has been considered one of the most suitable technologies for enabling access control. Users whose attributes match the access policy can access the raw data. ABE is mainly classified into two complementary forms, key policy ABE and ciphertext policy ABE. In CP-ABE, attributes are used to describe user attributes and access policies for these attributes are attached to the encrypted data. Due to its flexibility and expressiveness, CP-ABE has more applications in cloud storage access control. In this paper, we borrow CP-ABE as one of the components of our SACS design. Sanitizable Signatures: Sanitizable signatures (SS) are proposed to allow control over modifications to signed messages without invalidating the signature. SS is a form of digital signature through which a designated party (the sanitizer) can update permissible parts of the signed message. He introduced most of the security concepts in the purgable signature cipher proposed to hide the sanitizer's message signature pair. Many SS schemes have been proposed to meet different properties. SS provides the basis for the concept of sanitization in cryptography. Access Control Encryption: Access Control Encryption (ACE) was introduced to provide granular access control. ACE grants different rights to different users, not only regarding the messages they can receive, but also regarding the messages they can send. This includes an important sanitation feature. ACE can prevent corrupt senders from sending information to corrupt receivers. In ACE, the sanitizer uses its sanitizer key to run a specified randomization algorithm on the incoming ciphertext and then passes the result to the database server or recipients. Through sanitization, ACE ensures that no matter what the corrupted sender sends, what the recipient receives looks like a random encryption of a random message. In our SACS, the sterilization process does not require a sterilization switch from the authority. Only a valid recipient, who has been assigned a valid private key by the authority, can retrieve the message.

IV PROPOSED SYSTEM

We want to raise awareness about the concept of a sanitized access control system, or SACS, to prevent cloud storage by authors of untrusted data. Both data providers and recipients can set their own access restrictions using SACS. Like ABE, SACS allows any authorized recipient who possesses a private key and meets access requirements to decrypt the ciphertext. On the other hand, the sanitization functions provided by SACS prevent malicious data editors from creating ciphertexts that can be decrypted without using real private keys. The sanitizer will convert these ciphertexts into new ciphertexts that can only be decrypted by people with a legal private key, although it is possible that hostile data producers will create ciphertexts that anyone can understand. We present both our architecture and strategy for building SACS using the above technique. Additionally, we offer SACS installation. Additional Features SACS is based on the Attribute-Based Encryption (SA-BE) method. Thanks to SA-BE's ciphertext purification and malware protection, only valid private keys can be used to decrypt communications. This section includes a security model and formal algorithm definitions. SA-CS, which is based on SABE, uses CP-ABE, which is based on LSSS. The data editor encrypts regular data that has been encrypted using CP-ABE technology using a random key. The specific access policy for encrypted data must be verified before the sanitization process to maintain its integrity.

SYSTEM ARCHITECTURE:

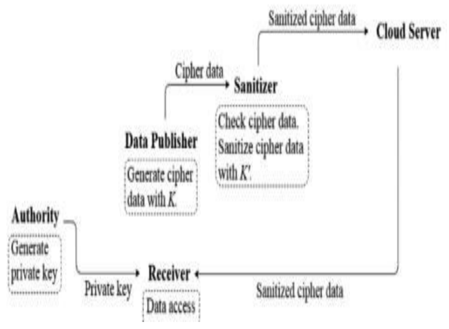

FIGURE 1. SACS architecture..

We show SACS architecture in **Fig. 1**, where five kinds of independent entities are involved. They are the authority, the data publisher, the sanitizer, the receiver and the cloud server.

- The authority manages and maintains the whole system. In SACS, we regard the authority as a trusted entity who holds the master secret key. The authority issues a unique private key to each receiver who registers into this system. Without loss of generality, we assume that the authority neither colludes with any other entities nor is compromised.
- The data publisher owns the plain data. He/she encrypts its plain data with an encryption key (e.g., K) and sets an access policy to deal with the encryption key. Then the data publisher sends the encrypted data (or cipher data) to the sanitizer. Actually, the publisher relies on this access policy to conduct data access control. Publishers are either honest or malicious. Both honest and malicious publishers execute the encryption operation on the plain data, but a malicious publisher might have extra behaviors, such as distributing the encryption key to some non-registered receivers. This incurs a failure of access control since some receivers can access the data without valid private key.
- The sanitizer is introduced to transform the original cipher data into the sanitized cipher data. Once getting cipher data from the data publisher, the sanitizer is instructed to do some specific processing on these cipher data. The processing includes two parts. One is to check whether the cipher data is under the claimed access policy and the other is to sanitize the cipher data with its encryption key K0 . Then the sanitizer sends the sanitized cipher data to the cloud server for storage. Such a sanitization operation on the cipher data is to prevent malicious publishers and invalid data access. The sanitizer is an honest party, which means it just executes the sanitization following the sanitizing algorithm but no malicious operations, such as replacing/modifying the cipher data. The sanitizer learns nothing about the plain data.
- The receiver wants to access the plain data. He/she can freely download the

cipher data that he/she is interested in from the cloud server. Prior to accessing the data, the receiver must register into the system and ask for a private key from the authority. When the registered receiver owns conditions satisfying the access policy, it is valid. Only valid receivers can access the plain data from the data publisher. Receivers will share neither their private keys nor the decrypted plain data with other entities. Here, we note that each receiver is unique.

- The cloud server provides a platform for cipher data storage. The cipher data stored in the cloud server can be acquired by any receivers. The cloud server just receives cipher data from the sanitizer and sends the cipher data to the receiver, while executes no computation operation. The cloud server will behave maliciously, e.g., delete the cipher data. Whether the cloud server is curious or not gives no effect on the security of SACS.

A. SANITIZED ATTRIBUTE-BASED ENCRYPTION

The SACS is based on a notion of Sanitized Attribute-based Encryption (SABE). SABE allows to sanitize the ciphertext and prevents malicious encryptors, such that only valid private keys can be used to obtain the message. This section gives formal algorithm definitions and security model.

- **Setup(; U).** The setup algorithm takes as input a security parameter and the number of universal attributes U. It returns system parameters Params and a master secret key msk.
- **KeyGen(S; msk;Params).** The key generation takes as input an attribute set S, the master secret key msk and the system parameters Params. It returns the private key skS of S.
- **Encrypt(P;M; Params).** The encryption algorithm takes as input an access policy P, a message M and the system parameter Params. It returns a ciphertext CT ¼ Enc½P; M; Params.
- **Sanitize(CT; Params).** The sanitization algorithm takes as input the system parameter Params and a ciphertext CT. It returns a sanitized ciphertext CT0 ¼ San½Params; CT.
- **Decrypt(CT0 ; skS; Params).** The decryption algorithm takes as input a sanitized ciphertext CT0 for P;M and a private key skS for S. If the attribute set S satisfies the access policy P, the decryption algorithm returns M ¼

Dec½CT0 ; skS; Params. Otherwise, it returns ?. Correctness. We give the correctness of Sanitized CP-ABE as follows. For all Params; msk; S; P such that the attribute set S satisfies the access policy P, if skS KeyGen (S; msk; Params) , CT Encrypt(P;M; Params) and CT0 Sanitize(CT; Params) , we have M ¼ DecryptðCT0 ; skS; ParamsÞ

V PERFORMANCE ANALYSIS

We present a performance analysis of our SACS protocol in terms of communication complexity and computation time. Because there are no comparable protocols in the literature, we only evaluate our protocol in the following analysis. To provide a fair analysis of communications, consider four stages excluding system initialization. We use jZpj, jGj, and jGT j to denote the size of an element in the sets Zp, G, and GT, respectively. Furthermore, the size of the feature set S is denoted by jSj and the length of the pseudorandom generator output is denoted by jPj. Table 2 summarizes the communication cost results. In the receiver registration phase, the communication cost is saved by the private key assigned by the authority to the receiver. The length of the private key is ð Þj 4 þ jSj Gj. In the encrypted data publishing stage, the communication cost mainly comes from the encrypted data, which is uploaded from the data editor to the sanitizer. The size of the encrypted data is ð Þj 1 þ m Gj þ 2jGT jþjPj, where m is the measure of a particular attribute set in the access policy. In the encrypted data sanitization phase, the communication cost is contributed by the sanitized encrypted data sent from the sanitizer to the cloud server. The size of the sanitized encrypted data is the same as the size of the encrypted data. In the data access stage, the recipient does not need to send anything to others, but needs to download the sanitized encrypted data stored on the cloud server. The communication cost is of course the volume of sanitized encrypted data. Here we do not consider the communication process of the request initiated by the recipient.

Then we consider different stages and measure the computation time at each stage. We implement the recipient registration phase, the encrypted data publishing phase, the encrypted data sanitization phase, and the data access phase. The system initialization phase is excluded because its calculation time is independent of the above-mentioned variable factors. fig. Numbers 2, 3, 4 and 5 show their corresponding calculation time, respectively. In the recipient registration stage, the calculation mainly comes from generating the private key, which must take the specified attribute as input. We then change the number of attributes and run the receiver registration phase algorithm accordingly. In the stage of publishing encrypted data, we focus on the computational time to generate the encrypted data, as we choose the simple format.

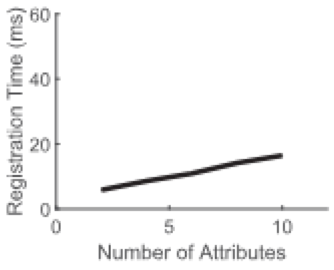

Fig no 2: Registration time.

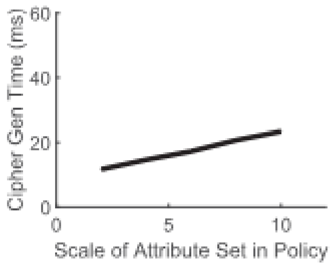

Fig no 3: Cipher gen time.

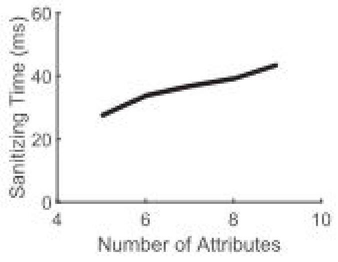

Fig no 4: Sanitizing time

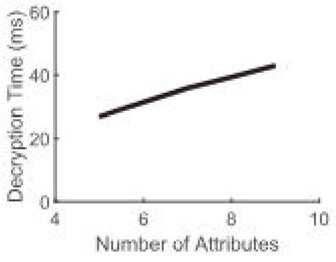

Fig no 5: Decryption time

Data with size of 10 KB. In terms of construction in the computational time will be associated with only the scale of a specific attribute set in the access policy and hence we execute trials with respect varying m. In the Cipher Data Sanitizing phase, the computational time is associated with both the scale of a specific attribute set in the access policy and the used attribute set. Here we choose to fix the former to be 10 and vary the latter. To test the computational time of the Data Access phase, we fix the scale of a specific attribute set in the access policy in the sanitized cipher data to be 10 and vary the number of attributes for the private key generated in the Receiver Registration phase.

Then we test the decryption time in the Data Access phase. Overall, these experiment results show that the computational time in each phase has a growth with increase of the specified variable factor although the amplitude variation is different.

CONCLUSION

We set out to study secure cloud storage in the presence of malicious data publishers, a very practical situation that has unfortunately not been studied before in the literature. In this setting, malicious data publishers create data following a specified access control policy, but the ciphertexts can be decrypted by unauthorized users without the need for valid keys. We design a system and its secure architecture to allow protection against these types of attacks. It is shown that our scheme is secure under the Diffie-Hellman q-parallel binary exponent assumption. We also provide an implementation of our system for performance analysis. We believe that this work will open the way for future research on cloud storage, as this idea is very practical. We note that this idea will further encourage the adoption of cloud storage in practice.

REFERANCES

[1] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proc. 13th ACM Conf. Comput. Commun. Secur., 2006, pp. 89–98.

[2] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in Proc. IEEE Symp. Secur. Privacy, 2007, pp. 321–334.

[3] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” in Proc. Int. Workshop Public Key Cryptogr., 2011, pp. 53–70.

[4] S. Berger et al., “Security intelligence for cloud management infrastructures,” IBM J. Res. Develop., vol. 60, no. 4, pp. 11:1–11:13, 2016.

[5] Secure access control for cloud storage. Accessed: Feb. 13, 2021. [Online]. Available: https://www.research.ibm.com/haifa/ projects/storage/cloudstorage/secure_access.s html.

[6] D. Boneh, C. Gentry, and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” in Proc. Annu. Int. Cryptol. Conf., 2005, pp. 258–275.

[7] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," ACM Trans. Inf. Syst. Secur., vol. 9, no. 1, pp. 1–30, 2006.

[8] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," IEEE Trans. Inf. Forensics Security, vol. 8, no. 12, pp. 1947–1960, Dec. 2013.

[9] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attributebased access control," IEEE Comput., vol. 48, no. 2, pp. 85–88, Feb. 2015.

[10] N. Attrapadung, B. Libert, and E. de Panafieu, "Expressive keypolicy attribute-based encryption with constant-size ciphertexts," in Proc. Int. Workshop Public Key Cryptogr., 2011, pp. 90–108.

[11] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attributebased solution for flexible and scalable access control in cloud computing," IEEE Trans. Inf. Forensics Security, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[12] Y. Wu, Z. Wei, and R. H. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing networks," IEEE Trans. Multimedia, vol. 15, no. 4, pp. 778–788, Jun. 2013.

[13] J. Hur, "Improving security and efficiency in attribute-based data sharing," IEEE Trans. Knowl. Data Eng., vol. 25, no. 10, pp. 2271–2282, Oct. 2013.

[14] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in Proc. Eur. Symp. Res. Comput. Secur., 2005, pp. 159–177.

[15] C. Brzuska et al., "Security of sanitizable signatures revisited," in Proc. Int. Workshop Public Key Cryptogr., 2009, pp. 317–336.

[16] V. Fehr and M. Fischlin, "Sanitizable signcryption: Sanitization over encrypted data (full version)," IACR Cryptol. ePrint Arch., vol. 2015, 2015, Art. no. 765.

[17] R. W. F. Lai, T. Zhang, S. S. M. Chow, and D. Schroder, "Efficient € sanitizable signatures without random Oracles," in Proc. Eur. Symp. Res. Comput. Secur., 2016, pp. 363–380.

[18] M. T. Beck et al., "Practical strongly invisible and strongly accountable sanitizable signatures," in Proc. Australas. Conf. Inf. Secur. Privacy, 2017, pp. 437–452.

[19] J. Camenisch, D. Derler, S. Krenn, H. C. Pohls, K. Samelin, and € D. Slamanig, “Chameleon-hashes with ephemeral trapdoors - and applications to invisible sanitizable signatures,” in Proc. Int. Workshop Public Key Cryptogr., 2017, pp. 152–182.

[20] N. Fleischhacker, J. Krupp, G. Malavolta, J. Schneider, D. Schroder, € and M. Simkin, “Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys,” in Proc. Int. Workshop Public Key Cryptogr., 2016, pp. 301–330.