Security Provisioning on a Cloud Platform

Abraham Jaison[1] Raji R Pillai[2]
[1]Student [2]Assistant Professor
[1,2]Department of Computer Science & Engineering
[1,2]Rajiv Gandhi Institute of Technology, Kottayam, India

Abstract— Cloud Computing, the ubiquitous form of computing, has almost taken over the world of computing. The cloud platform serves as a data repository which provides 'anywhere, anytime' access to the users. Users depend upon the cloud service providers and upload their data to the cloud which not only improves the availability of data but also the functionality. However many security threats have occurred and serious data leakages have taken place which poses a question on the integrity of data being stored in the cloud. This study segments the life cycle of data into different phases and reviews various existing technologies used to provide data security. Data security during transmission, storage, usage and destruction are discussed upon in this study.

Key words: Security, Cloud, Encryption, Storage, Destruction, Sensitive Data

I. INTRODUCTION

Sensitive data or sensitive information can be defined as any information that is protected against unwarranted disclosure. Access to sensitive information should be safeguarded and may be required for legal and/or ethical reasons. Retrieval of sensitive data for enterprises reduces the cost of providing users with personalized services as well as increasing their revenue. However with the intrusion of Cloud platform and big data platform in to the technological arena, the safekeeping of such sensitive data has also become an important area of concern. Having classified the life cycle of sensitive data into different stages, an analysis of different strategies are expounded herein.

Throughout the life of sensitive data four major phases have been identified to provide security. They are data submission or transmission, data storage on the cloud or big data platform, data usage and data destruction. If ample security is provided during all these four phases of data, security can be ensured.

The security provided during submission and/or transmission is crucial as many threats such as sniffing, eavesdropping etc can take place during those phases. When a user tries to upload a file or data from his/her personal computer, the file is moved unencrypted and taken to the server or rather the cloud platform. During the transmission from the personal computer to the cloud platform, we need to ensure the integrity of the file and ensure that even though it is received by the third party, the retrieved data should be inoperable. Once the data is uploaded and received at the storage server, it is the responsibility of the service provider to ensure the security of the data. Files stored in the cloud needs to be safe and has to be shielded from any kind of fissures. Security issues still pertain during the usage of such stored data and on the keeping of data on the platform. Users do not know how long the data will be kept in the system or even whether it exists after being deleted. Having segmented the life cycle of data into different sections, various algorithms and methodologies devised to address these concerns have been studied and are summarized in the following sections.

II. SECURING DATA TRANSMISSION AND STORAGE

One of the ever discussed topics of interest is the security provided to data during transmission. Numerous methodologies have been adopted to solve this and encryption strategies have always been a part of it. Encryption strategies are used almost in all phases of life for any kind of data. Various approaches have been developed and implemented to serve the purpose. Whenever the data is transmitted from the client side system to the cloud storage, steps must be taken to ensure the safe transmission of data. The data sent should also be kept safe even if attacks occur on the transmission line. Some of the major encryption strategies that have evolved over time and that could be used are outlined in this section.

A. Ciphertext-Policy Attribute Based Encryption

Ciphertext-Policy Attribute Based Encryption (CP-ABE) is a system for implementing diverse access control on protected/encrypted data. J. Bethencourt et. al. in [1] proposed this system to overcome the issue when a server storing the data is compromised in security. Formerly, the attributes in the Attribute Based Encryption systems described the encrypted data and embedded policies into user's keys, whereas in the system proposed in [2] the attributes are used to describe the credentials that describe the user's identity and the party that encrypts the data determines a policy that must be satisfied by the persons who want to decrypt. In this method, the private key of the user will be related to the attributes, which are expressed as strings. When a party encrypts a message, they specify an associated access structure over attributes. Decryption of a ciphertext for another user will only be possible if his attributes successfully pass through the access structure set for that particular ciphertext.

The CP-ABE algorithm also ensures collusion resistance by introducing a private key randomization technique. However the questions related to revoking the attributes assigned was not discussed upon in that study. In [3] S. Yi et. al. takes this forward and brings up a modification which enables the authority to revoke the attributes assigned to users efficiently by placing minimal loads on the authority and users. For this the Proxy Re-Encryption technique has been integrated with CP-ABE, which helps to delegate the herculean tasks to the proxy servers.

B. Fine Grained Access Control Systems Based On Attribute Based Encryption

The former methodologies placed minimal effort to address the issue of illegal key sharing. However, in [4] J. Li et. al. presented a way to implement scalable and fine-grained access control systems based on attribute based encryption

(ABE). In this work they proposed a modification to attribute based encryption to secure the access control in cloud computing by preventing the illegal key sharing among colluding users, which was not present in the former systems. One of the aims of this particular work is to identify the misbehaving users, who send out their secret keys.

Initially, for each file, the data owner defines and enforces access policies based on attributes. The owner then randomly chooses a key and encrypts the file using a standard algorithm. If another user wants to access the file, he should be issued a private key based on his attributes set by the authority. Once the user is allowed the access, a secret key with a long life span will be computed and sent to the user. The data owner also needs to update it to the cloud. A user can then download the ciphertext of his broadcast encryption and decrypt it and compute a message authentication code with the identities of the files to be retrieved. If a user is to be revoked, the data owner identifies a new key and uploads it to the cloud and deletes the former one that corresponds to that user. In order to trace an illegitimate user, whenever an unauthorized access attempt is detected, a third party auditor computes the hash values of identities for all authorized users and a match between the computed value and the possible attacker is taken.

C. Identity Based Encryption (IBE)

In all the described methods, the keys generated have to be distributed among users if the users are to effectively use the system. Identity-based encryption (IBE) [5], is a type of public-key encryption in which the users are provided with the facility to use the system without pre-distribution of the keys and the public key of a user is some unique information about the identity of the user (e.g. a user's email address). This can be used as the text-value of the name or domain name as a key or the physical IP address it translates to. Identity-based systems allow any party to generate a public key from a known identity value such as an ASCII string. A Private Key Generator (PKG) is invoked to generate the corresponding private keys. Initially the PKG publishes a master public key, and maintains the corresponding master private key. When the master public key is given, any party can generate a public key corresponding to the identity by merging the master public key with the identity value. Then in order to get the parallel private key, the party authorized to use the identity communicates with the PKG, which uses the master private key to generate the private key for identity. As a result, parties may encrypt messages with no prior distribution of keys between individual participants. This is extremely useful in cases where pre-distribution of authenticated keys is inconvenient or infeasible.

D. Proxy Re-Encryption (PRE)

Proxy re-encryption schemes [6] are cryptographic systems which allow third parties (proxies) to alter a ciphertext which has been encrypted for one party, so that it may be decrypted by another. The main aim of PRE is to securely enable the re-encryption of ciphertexts from one key to another, without depending on trusted parties. Proxy re-encryption schemes are similar to traditional symmetric or asymmetric encryption schemes, with the addition of two functions:

- Delegation - allows a message recipient (key holder) to generate a re-encryption key based on his secret key and the key of the delegated user.
- Transitivity - Transitive proxy re-encryption schemes allow for a ciphertext to be re-encrypted unlimited number of times.

E. Heterogeneous Proxy Re-Encryption (H-PRE)

All the former methodologies cover up only the data transmission and/or storage phases in the data life cycle. Heterogeneous Proxy re-encryption scheme proposed by X. Dong et. al. in [7] is a modified version of PRE where they have tried to enhance the security by combining three types of algorithms together. H-PRE focuses mainly on the security provided to the data on the big data platform particularly, since not many works have been focused to that direction. It tries to encompass all the phases in the data life cycle. It supports heterogeneous transformation from Identity Based Encryption to Public-Key Encryption. It transforms the cipher data that the owner uploads into ciphertext that the data user can decrypt using his or her own private key. It involves three types of algorithms – traditional identity-based encryption, re-encryption and traditional public key cryptosystems.

The data owner encrypts sensitive data using a local security plugin/advanced encryption standard and then the PRE algorithm is used to encrypt the symmetric key of the data. Should the data be shared with other users, then the owner must authorize the user and generate a PRE key that is to be stored in the server. Then the encrypted data is uploaded to the big data platform where the server re-encrypts and transforms the original cipher with the PRE key. PRE ciphertext is then generated which can be encrypted by the authorized users. A data user who wants to use the data on the big data platform sends requests to the platform and if available it is made available for downloading.

The H-PRE algorithm is capable of handling the data submission, storage and extraction operations on a big data platform. The PRE computational work is the main overhead involved in this method, which is transferred to the powerful big data platform, in this case, to improve the user experience.

III. SECURING DATA USAGE/COMPUTATION

In the cloud environment, data is used online using various hypervisors or virtual machine monitors. Therefore, apart from securing the transmission and storage of data, securing the computation or usage of data on the cloud platform is also to be given prime importance. A hypervisor (or Virtual Machine Monitor) is a piece of software that manages the sharing of a hardware platform among multiple guest systems. Hypervisors are increasingly used these days ranging from PCs to web servers and data centers. If the hypervisor is compromised, then the entire system is compromised, since all the virtual machines run on top of the hypervisors, and hence it is vital to ensure the security of virtual machine monitors (VMM) or the hypervisors. This section tries to brief the concepts related to securing the data during computation on the peripheral platform.

A. HIMA

Virtualization provides strong isolation for users to run their programs. However the runtime protection provided is limited. In [8] A. M. Azab et. al. proposed a hypervisor-based agent that measures the integrity of guest virtual machines running on top of the hypervisor (HIMA). In addition to providing strong isolation and time of check to time of use consistency (TOCTTOU), it also supports two complementary techniques – active monitoring of critical guest events and guest memory protection. Active monitoring of critical guest events monitors all events that change guest program's layout so that the measurement can be performed accordingly and ensures to be up-to-date. Guest memory protection guarantees capturing any attempt to modify measured programs or to bypass the measurement.

HIMA performs monitoring of events by checking the intercepted guest events such as system calls, interrupts and exceptions. It ascertains that the integrity measures are refreshed whenever the program layout in a guest VM changes. Guest memory protection provided by HIMA ensures that no bypassing especially related to integrity measurement of user programs, can be done without the knowledge of it.

B. HyperSentry

HyperSentry, proposed by A. M. Azab et. al. in [9] presents a structure to ensure integrity of an active hypervisor. HyperSentry relies on Trusted Computing Base properly isolated from the highest privileged software. Unlike HIMA, HyperSentry is triggered by an out-of-band communication channel that is out of the control of the system's CPU and consequently the highest privileged software. It is in the System Management Mode to protect is base code and critical data.

HyperSentry ensures that it maintains a stealthy invocation without alerting the users, so that it can silently monitor the activities without the knowledge of the potential attacker. The security given by hypersentry is ascertained to be high as it provides the following five qualities: stealthy invocation, verifiable behavior, deterministic execution, in-context privileged measurement and attestable output. Remote users can rely on HyperSentry as it has a very small code base and does not possess any interaction with any other software.

IV. SECURING DATA DESTRUCTION

In the context of cloud computing, data is stored in an external storage space, and in order to provide reliability and availability, the data is replicated and stored in multiple locations. Whether they are sensitive or insensitive, there needs to be a proper destruction strategy for removing these data, which is necessary to ensure that the data is safe throughout and after its life. Various methodologies have been proposed to address this issue and some are outlined below.

A. SafeVanish

SafeVanish is a scheme proposed by L. Zeng et. al. in [10] to prevent hopping attacks and sniffer attacks on the data stored in cloud. In order to avoid sniffing attack and increase the hopping attack cost, SafeVanish extends the length range of the key shares and apply the public-key cryptosystem. As the length of range of key shares is increased substantially, an attacker is required to monitor in a very large range. When such kind of operations are performed the bandwidth and storage space requirement grows significantly.

To overcome the sniffing attack, strict isolation or protection is to be given for the user's traffic to the DHT. Hence a public key cryptography can be used to formulate a solution to this. The sender's private key is to be used initially to encrypt the key shares, using the RSA encryption algorithm, before sending them to the DHT node. Following which, the receiver's public key can be used to encrypt the key shares to complete the encryption operation.

B. Secure Self Destructing Scheme for Electronic Data

Secure self-destructing scheme for electronic data, proposed by G. Wang et. al. in [11] attempts to protect the sensitive data of a user by automatically destroying it after a period of time, without any explicit action by the user or any third party while ensuring that the data should be accessible to the authorized users before the expiration. This scheme ensures that the process just extends the key space of the traditional symmetric encryption scheme and does not cause any changes to the encryption scheme. Security is achieved by first encrypting the data, then associating and extracting the ciphertext, and finally distributing both the decryption key and a part of the ciphertext into the distributed hash table (DHT) network. A DHT is a data structure for storing, managing and querying data in distributed systems or P2P networks.

In this methodology, it is not just the original message that is encrypted but the ciphertext is also involved and extracted. Every block is related to each other and hence the original message cannot possibly be recovered without the complete ciphertext and decryption key. A major advantage possessed by this scheme is that it does not depend heavily on the encryption algorithm. DHT is a very safe network as it removes older data after a time period so that both decryption key and the ciphertext are also destructed, and also due to its huge size, geographic distribution and decentralization, the attacks on DHT networks are much more challenging and experimental results show that it is almost infeasible.

V. CONCLUSION

An overview of the basic encryption strategies and some of the methodologies used to provide the security for the data while being used in the cloud platform and also during destruction have been briefed in this paper. The sole aim was to find out how effectively can data sharing be done securely via the cloud platform. The idea of integration as explained by the H-PRE algorithm seems to be a good breakthrough for securing the data throughout its life cycle. The secure usage can be affirmed by securely downloading it to the private space of a user with enough protection to the hypervisor. The secure self-destruction scheme provides protection to the data that is to be deleted. Hence throughout the entire life of data, it can be safeguarded against any unwarranted disclosures.

REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-policy attribute-based encryption, in Proc. IEEE Symposium on Security and Privacy, Oakland, USA, 2007, pp. 321–334.
[2] S. Yu, C. Wang, K. Ren, and W. Lou, Attribute based data sharing with attribute revocation, in Proc. 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 2010, pp. 261–270.
[3] S. Yu, C. Wang, K. Ren, and W. Lou, Attribute based data sharing with attribute revocation, in Proc. 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 2010, pp. 261– 270.
[4] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, L. Tang, and Y. Tang, Fine-grained data access control systems with user accountability in cloud computing, in Proc. 2nd Int. Conf. on Cloud Computing, Indianapolis, USA, 2010, pp. 89–96.
[5] D. Boneh and M. Frankliny, Identity-Based Encryption from the Weil Pairing, in SIAM Journal of Computing, Vol. 32, No. 3, 2003, pp. 586-615
[6] X. Wang, X. Yang, and M. Zhang, Proxy Re-encryption Scheme from IBE to CBE, in First International Workshop on Database and Technology Application, IEEE Computer Society, 2009, pp. 99-102
[7] X. Dong, R. Li, H. He, W. Zhou, Z. Xue and H. Wu, Secure Sensitive Data Sharing on a Big Data Platform, in Tsinghua Science and Technology Journal, ISSN 1007-2014 08/11, 2015, pp.72-80
[8] A. M. Azab, P. Ning, E. C. Sezer, and X. Zhang, HIMA: A hypervisor-based integrity measurement agent, in Proc. 25th Annual Computer Security Applications Conf., Hawaii, USA, 2009, pp. 461–470.
[9] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky, HyperSentry: Enabling stealthy in-context measurement of hypervisor integrity, in Proc. 17th ACM Conference on Computer and Communications Security, Chicago, USA, 2010, pp. 38–49.
[10] L. Zeng, Z. Shi, S. Xu, and D. Feng, Safevanish: An improved data self-destruction for protecting data privacy, in Proc. 2nd Cloud Computing International Conf., Indianapolis, USA, 2010, pp. 521–528.
[11] G. Wang, F. Yue, and Q. Liu, A secure self-destructing scheme for electronic data, Journal of Computer and System Sciences, vol. 79, no. 2, pp. 279–290, 2013.

