Cloud Data Security Using Homomorphic Encryption

NAMPALLY PRASHANTH,
Assistant Professor,
Department of Computer Science and Engineering,

MOHAMMED MOQUEED AHMED,
Assistant Professor,
Department of Electronics and Communications Engineering,

Siddhartha Institute of Technology and Sciences,
Narapally, Hyderabad, Telangana – 500 088.

Abstract

The phenomena of cloud computing is vast and diversified. Users can save vast amounts of data to the cloud for later use. Data security, privacy, confidentiality, integrity, and authentication are just a few of the security concerns that must be handled. The majority of cloud service providers keep data in plaintext, and users must employ their own encryption technique to safeguard their data if necessary. When processing data, it must be decrypted. This work focuses on storing data in encrypted format in the cloud using fully homomorphic encryption. The data is kept in Amazon Web Service's (AWS) public cloud's DynamoDB. In the public cloud, the user's computation is performed on encrypted data. When the results are needed, they may be downloaded to the client system. Users' data is never kept in unencrypted on the public cloud in this situation.

1. Introduction

Cloud computing security is a serious problem. There is a tremendous push to ensure infrastructure security at the network, host, application, and data levels. Data is related with each level, such as network, host, and application. The security of cloud data at rest is the topic of this research. Several technologies are used in cloud computing.

Security concerns about many types of assaults involving various technologies must be addressed. Some of the security concerns in cloud computing are as follows: Data availability is a critical security problem. It must be made available to the user whenever it is necessary. In addition, the user must have control over its data.

When service from another cloud service provider is requested, an availability issue must be addressed. Currently, there are three significant concerns to availability. The first danger is a network-based cyber-attacks. The second issue is the availability of cloud service providers, and the third is the backup of stored data by cloud service providers.

There is a need for effective and efficient approaches for controlling access to, authenticating, and authorising sensitive data. Data remanence is a problem that occurs when data is accessible

to an unauthorised entity after erasure. The complete process of data generation through destruction is referred to as the data security lifecycle. When it comes to data destruction, extreme caution must be exercised.

Third-Party Control- The user data is managed by a cloud service provider. Access from a third party may result in the disclosure of sensitive information and trade secrets. Corporate espionage is also a significant risk. It should also not force users to rely solely on a single cloud service provider.

2. Literature survey

When we transmit data to the cloud, we use typical encryption methods to safeguard it, but when we want to run computations on data hosted on a remote server, we need the cloud provider to have access to the raw data, which it will decode. As we all know, the need for data privacy and algorithms to handle corporate information has grown dramatically over the previous few decades. To accomplish this, technologies such as data encryption techniques and tamper-resistant hardware are utilised.

Cloud computing adoption is definitely a strategic path for many businesses. The confluence of low-cost computing, ubiquitous mobility, and virtualization technologies has resulted in a platform for more agile and cost-effective corporate applications and IT infrastructure. The cloud forces the unique and careful use of security measures, resulting in a requirement for best practises in security programmes and governance regimes. CloudSecurityandPrivacy is a handbook for individuals who are struggling to develop cloud security.

Even though your data is encrypted while being saved, keys are frequently stored with it. In order to execute calculations on the encrypted data and therefore keep the key safely, an end-to-end encryption technique has been presented as a viable option for data storage in the cloud. Somewhat Homomorphic Encryption is a completely homomorphic encryption approach that is compact, semantically safe, uses a much smaller public key, and can encrypt integer plaintexts rather than single bits. It also has reduced expansion and processing complexity.

Due to their high computational difficulties and large message expansions, existing completely homomorphic systems aren't genuinely feasible. This study presents a Somewhat Homomorphic public key encryption technique, which may be thought of as a version of the scheme proposed by Van Dijk et al, but with a wider message space. The suggested technique is small, semantically safe, and capable of encrypting integer plaintexts rather than single bits, with lower message expansion and computational complexity.

Assume you wish to distribute the ability to process your data but not the data itself. They demonstrate that this separation is possible: we provide a "completely homomorphic" encryption technique that keeps data private while allowing a worker without the secret

decryption key to compute any (still encrypted) outcome of the data, even when the data's function is exceedingly complicated. In other words, a third party can process data without seeing it. This aids cloud computing's compatibility with privacy, among other reasons.

3. Methodology

The method used in the proposed approach is fully homomorphic. This technique is a reduced and efficient variant that is used to secure user data on the AWS public cloud. A secret key is represented by (J, K), whereas a public key is represented by (P0, P1). As user input, the number N to be encrypted is accepted.

- **Client Machine-** Through the client machine, requests are made to obtain data from the cloud server.
- **Login-** The login component assists the user with logging into the system without using the right username and password, which are confirmed by the server before the user is granted access to the system.
- **Key Selection-** For data encryption and decryption, this component chooses a key depending on the user who has logged into the system.
- **Query-** After logging in, the user will pick the operation to be conducted.
- **Computations -** Computations are conducted and handed on to the encryption component to be stored based on the operation selected.
- **Encrypt and store-** This component encrypts the data entered by the user or the data computed by the system before storing/ updating the value in the cloud database.
- **Retrieve and decrypt-** This component obtains and decrypts the data that the user requires from the cloud database and delivers it to the user on the client machine.
- **AWS Cloud (DynamoDB) -** This is the cloud database where all of the data is stored and accessed by the login module for user verification, the Key selection component for retrieving the keys stored in the database, the encryption component for storing data in encrypted format, the decryption component for retrieving and decrypting the data, and computation for performing operations on the user data as per requirement or the query fired.

The Eclipse IDE for Java EE Developers may be used to connect to the AWS DynamoDB service. This allows the user to log in using his credentials and then execute operations on their data according to their needs. When the user has completed all of the tasks, he or she can choose to depart the system.

The stages involved in the implementation are listed below.

Step 1: On AWS, create a DynamoDB instance.

Step 2: Create database tables that follow the correct schema.

Step 3: Get the AWS credentials and set up access controls.

Step 4: Download and install the Eclipse Kepler version, as well as the Java SDK.

After installing the AWS SDK on the Eclipse framework, the user has access to all of the required packages.

Step 5: Follow the instructions in the 23AWS SDK.

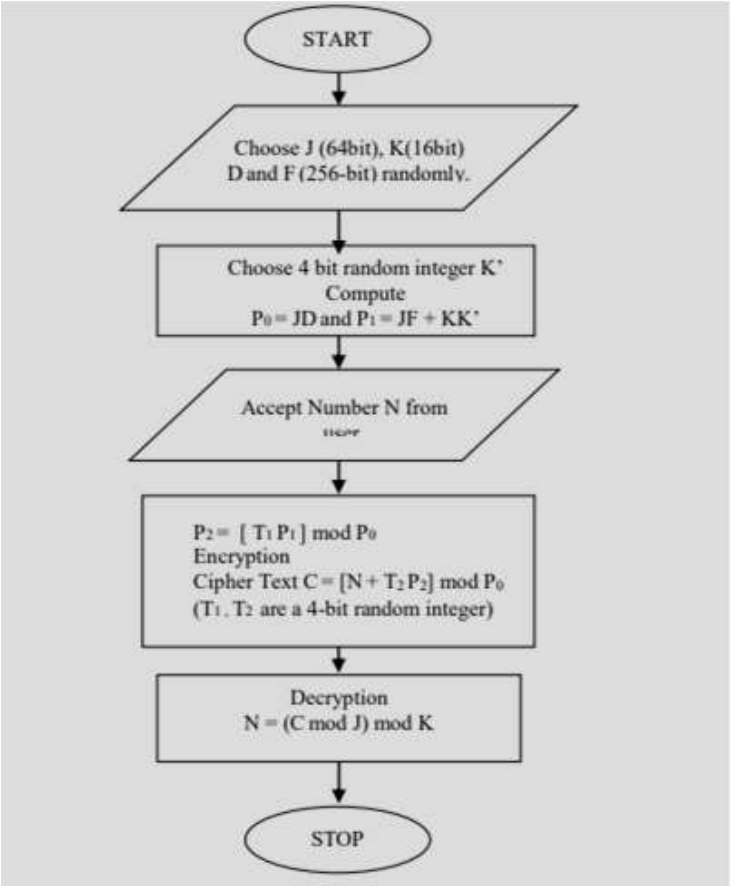

Fig.1 Flowchart of Fully Homomorphic Encryption Scheme

The Java code that interacts with the DynamoDB database. This eclipse platform is used to execute it. This platform handles all of the necessary interaction, such as data operations like addition, subtraction, and check balances in the database. The user logs in to the system using the given interface and then conducts functions supplied. Only the privileges granted to him by the database owner are available to the user.

4. Result and discussion

On DynamoDB, two tables are generated. The homomorphic encryption algorithm is used to store the balance. This encrypted balance can be added to and subtracted from by the user. The balance may be checked in plaintext by the user.

Fig. 2 Database on DynamoDB

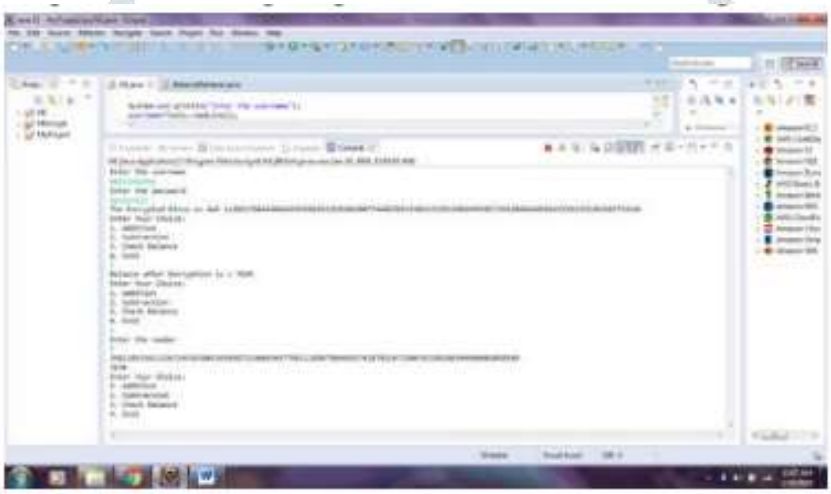

Fig. 3 Executions at Client

5. Conclusion

Homomorphic Encryption will provide cloud storage a new dimension. It ensures data secrecy since data is never exposed in plain text at any level. The suggested technique is a reduced, efficient variant that has been used on the AWS public cloud. The suggested algorithm has a wide range of applications, including online auctioning, medicinal applications, and commercial applications.

For effective data processing, research into lowering the amount of cypher text is required. It is also necessary to develop various methods for finding and querying encrypted data under the FHE protocol.

References

1. Tebaa, M.; El Hajji, S.; El Ghazi, A., "Homomorphic encryption method applied to Cloud Computing," in Network Security and Systems (JNS2), vol., no., pp.86-89, 20-21 April 2012.
2. Mather, Tim, Subra Kumaraswamy, and Shahed Latif. Cloud security and privacy: an enterprise perspective on risks and compliance. " O'Reilly Media, Inc.", 2009
3. Samyak Shah, Yash Shah, Janika Kotak, “Somewhat Homomorphic Encryption Technique with its Key Management Protocol”, Dec 14, Volume 2 Issue 12 , International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), ISSN: 2321-8169, PP: 4180 - 4183
4. Ramaiah, Y. Govinda, and G. Vijaya Kumari. "Efficient public key homomorphic encryption over integer plaintexts." Information Security and Intelligence Control (ISIC), 2012 International Conference on. IEEE, 2012.
5. Gentry, Craig. "Computing arbitrary functions of encrypted data." Communications of the ACM 53.3 (2010): 97-105.
6. Atayero, Aderemi A., and Oluwaseyi Feyisetan. "Security issues in cloud computing: The potentials of homomorphic encryption." Journal of Emerging Trends in Computing and Information Sciences 2.10 (2011): 546-552.
7. Catteddu, Daniele, and Giles Hogben. "Cloud computing." Benefits, Risks and Recommendations for Information Security/European Network and Information Security Agency, ENISA (November 2009) (2009).
8. Deyan Chen; Hong Zhao, "Data Security and Privacy Protection Issues in Cloud Computing," in Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on , vol.1, no., pp.647-651, 23-25 March 2012
9. Pearson, Siani. "Taking account of privacy when designing cloud computing services." Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. IEEE Computer Society, 2009.
10. Rivest, Ronald L., Len Adleman, and Michael L. Dertouzos. "On data banks and privacy homomorphisms." Foundations of secure computation 4.11 (1978): 169-180.