Homomorphic Encryption: What Is It and How Is It Used

Anastasios Arampatzis

Every day, organizations handle a lot of sensitive information that needs to be encrypted both when it is stored (data at rest) and when it is being transmitted (data in transit).

The problem with encrypting data is that sooner or later, you have to decrypt it. And decrypting data makes it vulnerable to hackers. You can keep your cloud files cryptographically scrambled using a secret key, but as soon as you want to actually do something with those files—anything from editing a word document or querying a database of financial data—you have to unlock the data and leave it vulnerable. Homomorphic encryption, an advancement in the science of cryptography, changes that.

What is Homomorphic Encryption?

The purpose of homomorphic encryption is to allow computation on encrypted data. Thus data can remain confidential while it is processed, enabling useful tasks to be accomplished with data residing in untrusted environments. In a world of distributed computation and heterogeneous networking this is extremely valuable.

A homomorphic cryptosystem functions similarly to other types of public encryption in the sense that it utilizes a public key to encrypt data and only permits the person with the corresponding private key to access the unencrypted data. Its distinctiveness, however, lies in its use of an algebraic system that enables a range of computations (or operations) on the encrypted data.

In the realm of mathematics, the term homomorphic refers to the conversion of one data set into another while maintaining the relationships among elements in both sets. The word originates from the Greek terms for "same structure." Because the data in a homomorphic encryption scheme preserves the same structure, identical mathematical operations conducted on encrypted or decrypted data will yield the same results.

In real-world applications, most homomorphic encryption systems function optimally with data expressed as integers and when using operational functions like addition and multiplication. This allows the encrypted data to be analyzed and manipulated as if it's in plaintext format without actually decrypting it. The encrypted data can be computed and processed to get an encrypted answer, but only the private key owner can decrypt the ciphertext and understand what it means.

Why use homomorphic encryption

Organizations can use traditional encryption methods to secure sensitive data on cloud environments. But if they need to investigate or validate encrypted data in the cloud, they would need to either decrypt the data or download it and decrypt it. The first option can lead to security problems and the second can be costly and time-consuming.

That's where the true value of homomorphic encryption comes to the fore. Homomorphic encryption enables organizations to share private data to be evaluated securely without jeopardizing privacy. Homomorphic encryption gives organizations the ability to perform mathematical operations on encrypted data without exposing the data itself.

With homomorphic encryption, the cloud service provider has access only to encrypted data and can perform computations on it without decrypting it. They can then return the encrypted results to the owner of the private data who can decrypt it with a private key.

Categories of Homomorphic Encryption

Homomorphic encryption is classified into three categories. These categories are distinguished based on the kind and frequency of mathematical computations that can be executed on the ciphertext. The three categories are outlined below:

Partially Homomorphic Encryption

Partially Homomorphic Encryption (PHE) permits only specific mathematical operations to be executed on encrypted values. This implies that only one operation, either addition or multiplication, can be carried out indefinitely on the ciphertext. Multiplicative PHE is the cornerstone of RSA encryption, a common method used to establish secure connections via SSL/TLS.

Somewhat Homomorphic Encryption

A Somewhat Homomorphic Encryption (SHE) system supports a specific operation (either addition or multiplication) up to a particular complexity level, but these operations can only be carried out a fixed number of times.

Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) holds great promise for harmonizing functionality with privacy by aiding in maintaining data security while keeping it accessible simultaneously. Evolved from the SHE system, FHE can use both addition and multiplication operations indefinitely, enhancing the efficiency of secure multi-party computation. Unlike other types of homomorphic encryption, it can manage arbitrary computations on your ciphertexts.

The aim of FHE is to enable individuals to utilize encrypted data to perform valuable operations without requiring the encryption key. Specifically, this concept has potential uses for enhancing cloud computing security. If you intend to store encrypted data in the cloud but want to avoid the risk of a hacker compromising your cloud account, FHE offers you a method to retrieve, search, and modify your data without granting the cloud provider access to your data.

Security of Fully Homomorphic Encryption

The security of the homomorphic encryption schemes is based on the Ring-Learning With Errors

(RLWE) problem, which is a hard mathematical problem related to high-dimensional lattices. A great number of peer-reviewed research confirming the hardness of the RLWE problem gives us confidence that these schemes are indeed at least as secure as any standardized encryption scheme.

Applications of Fully Homomorphic Encryption

Craig Gentry mentioned in his graduation thesis that "Fully homomorphic encryption has numerous applications. For example, it enables private queries to a search engine—the user submits an encrypted query and the search engine computes a succinct encrypted answer without ever looking at the query in the clear. It also enables searching on encrypted data—a user stores encrypted files on a remote file server and can later have the server retrieve only files that (when decrypted) satisfy some Boolean constraint, even though the server cannot decrypt the files on its own. More broadly, fully homomorphic encryption improves the efficiency of secure multi party computation."

Researchers have already identified several practical applications of FHE, some of which are discussed herein:

- **Securing Data Stored in the Cloud**. Using homomorphic encryption, you can secure the data that you store in the cloud while also retaining the ability to calculate and search ciphered information that you can later decrypt without compromising the integrity of the data as a whole.
- **Enabling Data Analytics in Regulated Industries**. Homomorphic encryption allows data to be encrypted and outsourced to commercial cloud environments for research and data-sharing purposes while protecting user or patient data privacy. It can be used for businesses and organizations across a variety of industries including financial services, retail, information technology, and healthcare to allow people to use data without seeing its unencrypted values. Examples include predictive analysis of medical data without putting data privacy at risk, preserving customer privacy in personalized advertising, financial privacy for functions like stock price prediction algorithms, and forensic image recognition.
- **Improving Election Security and Transparency**. Researchers are working on how to use homomorphic encryption to make democratic elections more secure and transparent. For example, the Paillier encryption scheme, which uses additional operations, would be best suited for voting-related applications because it allows users to add up various values in an unbiased way while keeping their values private. This technology could not only protect data from manipulation, it could allow it to be independently verified by authorized third parties.

Limitations of Fully Homomorphic Encryption

There are currently two known limitations of FHE. The first limitation is support for multiple users. Suppose there are many users of the same system (which relies on an internal database that is used in computations), and who wish to protect their personal data from the provider. One solution would be for the provider to have a separate database for every user, encrypted under that user's public key. If this database is very large and there are many users, this would quickly become infeasible.

Next, there are limitations for applications that involve running very large and complex algorithms homomorphically. All fully homomorphic encryption schemes today have a large computational overhead, which describes the ratio of computation time in the encrypted version versus computation time in the clear. Although polynomial in size, this overhead tends to be a rather large polynomial, which increases runtimes substantially and makes homomorphic computation of complex functions impractical.

Implementations of Fully Homomorphic Encryption

Some of the world's largest technology companies have initiated programs to advance homomorphic encryption to make it more universally available and user-friendly.

Microsoft, for instance, has created SEAL (Simple Encrypted Arithmetic Library), a set of encryption libraries that allow computations to be performed directly on encrypted data. Powered by open-source homomorphic encryption technology, Microsoft's SEAL team is partnering with companies like IXUP to build end-to-end encrypted data storage and computation services. Companies can use SEAL to create platforms to perform data analytics on information while it's still encrypted, and the owners of the data never have to share their encryption key with anyone else. The goal, Microsoft says, is to "put our library in the hands of every developer, so we can work together for more secure, private, and trustworthy computing."

Google also announced its backing for homomorphic encryption by unveiling its open-source cryptographic tool, Private Join and Compute. Google's tool is focused on analyzing data in its encrypted form, with only the insights derived from the analysis visible, and not the underlying data itself.

Finally, with the goal of making homomorphic encryption widespread, IBM released its first version of its HElib C++ library in 2016, but it reportedly "ran 100 trillion times slower than plaintext operations." Since that time, IBM has continued working to combat this issue and have come up with a version that is 75 times faster, but it is still lagging behind plaintext operations.

(*This post has been updated. It was originally posted on January 1, 2020.*)

Related posts

- Using Blockchain for Searchable Encryption as a Service
- How is Diffie-Hellman Key Exchange Different than RSA?
- Budget for Encryption Increasing Over Time, Reveals Survey
- Is the War on Encryption a Fight Between Privacy and Safety?