Elements of Algebraic Structures

Homomorphic Encryption

An Introduction from a Point of View of Algebra

Sandeep Chatterjee

M.Tech. Computer Science

19th April 2024

Indian Statistical Institute

Abstract

In this presentation, a gentle introduction to homomorphic encryption is presented, a cutting-edge cryptographic technique with profound implications for data privacy and computational security. Through an exploration from an algebraic perspective, we will delve into the fundamental principles underlying homomorphic encryption, elucidating its mechanisms and capabilities. By examining its algebraic foundations, we aim to demystify this powerful cryptographic tool and shed light on its practical applications.

In this write-up, we explore the theoretical underpinnings and practical implementations of homomorphic encryption, beginning with an overview of algebraic homomorphisms and their relevance to cryptography. It traces the historical development of homomorphic encryption, from the pioneering work of Rivest, Adleman, and Dertouzos to modern schemes like the Paillier cryptosystem. The challenges and advancements in fully homomorphic encryption, exemplified by Craig Gentry's seminal work, are discussed, highlighting its transformative potential despite computational complexities. This paper concludes by emphasizing the ongoing research in fully homomorphic encryption, reflecting its significance in contemporary cryptography.

Click for the presentation

Contents

0.1 Motivation

Traditional encryption methods excel at data security but present a fundamental challenge to computational processes. Encrypting all inputs using conventional techniques renders subsequent operations ineffective due to the loss of underlying structure. This underscores the pressing demand for inventive approaches harmonizing data privacy and computational functionality. Enter homomorphic encryption, a revolutionary cryptographic paradigm empowering computations directly on encrypted data sans decryption, offering a transformative solution to this problem

In today's interconnected digital landscape, preserving privacy while harnessing the advancements of machine learning and algorithms. Homomorphic encryption offers a groundbreaking solution by allowing computations on encrypted data without compromising its confidentiality. Consider the scenario of spam classification in email systems: With homomorphic encryption, service providers can analyze encrypted email content to detect spam, safeguarding user privacy while effectively filtering unwanted messages. This exemplifies the transformative potential of homomorphic encryption in enabling secure, privacy-preserving data processing across various domains.

Chapter 1

Preliminaries

1.1 Algebraic Homomorphisms

1.1.1 Definition (Group Homomorphism)

Let $(G, *)$ and $(H, \diamond)$ be groups. The map $\phi : G \to H$ is a homomorphism if and only if:

$$\phi(x * y) = \phi(x) \diamond \phi(y) \quad \forall x, y \in G$$

1.1.2 Definition (Ring Homomorphism)

Let R and S be rings with addition and multiplication. The map $\phi : R \to S$ is a homomorphism if and only if:

1. ϕ is a group homomorphism on the additive groups $(R, +)$ and $(S, +)$:

$$\phi(a + b) = \phi(a) + \phi(b), \quad \forall a, b \in R$$

2. ϕ preserves multiplication:

$$\phi(xy) = \phi(x) \cdot \phi(y), \quad \forall x, y \in R$$

1.2 Some Useful Properties

1. We have the inclusion homomorphism $\iota : \mathbb{Z} \to \mathbb{Q}$, which sets $\iota(n) = n$. This map clearly preserves both addition and multiplication.

2. Consider the map $\phi : \mathbb{Z} \to \mathbb{Z}_n$ sending k to k. We've seen that this is a homomorphism of additive groups, and can easily check that multiplication is preserved. Indeed,

$$\phi(a) = \phi(1 + 1 + \cdots + 1) = \phi(1) + \phi(1) + \cdots + \phi(1) = a\phi(1) = a.$$

Notice that every element in $\mathbb{Z}$ can be written as a sum of many copies of 1. Then we were able to determine what the homomorphism does simply by knowing $\phi(1)$.

3. The evaluation map e_k is a function from $R[x]$ to R. For any polynomial $f \in R[x]$ and $k \in R$, we set $e_k(f) = f(k)$. This is a ring homomorphism! Let $f(x) = a_n x^n + \cdots + a_0 x^0$ and $g(x) = b_n x^n + \cdots + b_0 x^0$, where the a_i, $b_i \in R$. (We'll also allow leading coefficients to be zero in order to make it easy to add f and g formally.) We then check the ring homomorphism conditions: since we know that e_k is an additive homomorphism, we only need to check that it is multiplicative on monomials. But that's easy:

$$e_k((a_n x^n)(b_m x^m)) = e_k(abx^{n+m}) = abk^{n+m} = e_k(a_n x^n)e_k(b_m x^m).$$

1.3 Homomorphisms and Cryptography

Homomorphisms play a significant role in encryption technique to enable computations to be performed directly on encrypted data without the need for decryption, thereby preserving the confidentiality of sensitive information. Homomorphic encryption finds applications in various domains, including e-cash, e-voting, private information retrieval, and cloud computing, where privacy-preserving computation is paramount.

For over 30 years, achieving fully homomorphic encryption, has been a major goal in cryptography. The breakthrough came in 2009 when Craig Gentry proposed the first-ever fully homomorphic encryption system. Prior to this milestone, encryption systems capable of preserving computations under a single operation had been utilized for decades, laying the groundwork for the development of more advanced cryptographic protocols and systems.

Chapter 2

History, Developement & Implementation

2.1 On Data Banks and Privacy Homomorphisms

- Rivest, Adleman, and Dertouzos, Rivest et al. 1978
- Introduced the idea of ”Privacy Homomorphisms”
- Introduced four possible encryption functions (RSA was one of them)

They illustrated its practical implications, such as a loan company encrypting sensitive data stored in a time-sharing service, highlighting the feasibility of privacy-preserving techniques.

2.2 Blind Signatures for Untraceable Payments

- David Chaum,
- Calls for a payment system with:
  - Anonymity of payment , Proof of payment
- Analogy to secure voting
  - Place vote in a carbon envelope The signer can then sign the envelope, consequently signing the vote without ever knowing what the vote is
- Paper doesn't gives any solution but introduces the problem with analogies highlights the various use cases. Although no mention of a private homomorphism, the paper helps introduce the need for secure voting as well as the relationship between e-cash and e-voting.

2.3 Example: The RSA Cryptosystem

Definition (RSA): Let $n = pq$ where p and q are primes. Pick a and b such that $ab \equiv 1 \pmod{\phi(n)}$. n and b are public while p, q, and a are private.

$$e_K(x) = x^b \mod n$$

$$d_K(y) = y^a \mod n$$

The Homomorphism: Suppose x_1 and x_2 are plaintexts. Then,

$$e_K(x_1)e_K(x_2) = x_1^b \cdot x_2^b \mod n = (x_1x_2)^b \mod n = e_K(x_1x_2)$$

2.4 ElGamal Cryptosystem

Definition (ElGamal): Let p be a prime and pick $\alpha \in \mathbb{Z}_p^*$ such that α is a generator of $\mathbb{Z}_p^*$. Pick a and β such that $\beta \equiv \alpha^a \pmod{p}$. p, α, and β are public; a is private. Let $r \in \mathbb{Z}_{p-1}$ be a secret random number. Boneh and Shoup 2023 Then, $e_K(x, r) = (\alpha^r \mod p, x \cdot \beta^r \mod p)$

The Homomorphism: Let x_1 and x_2 be plaintexts. Then,

$$\begin{aligned} e_K(x_1, r_1) \cdot e_K(x_2, r_2) &= (\alpha^{r_1} \mod p, x_1 \cdot \beta^{r_1} \mod p) \cdot (\alpha^{r_2} \mod p, x_2 \cdot \beta^{r_2} \mod p) \\ &= (\alpha^{r_1} \cdot \alpha^{r_2} \mod p, x_1 \cdot \beta^{r_1} \cdot x_2 \cdot \beta^{r_2} \mod p) \\ &= (\alpha^{r_1+r_2} \mod p, x_1 \cdot x_2 \cdot \beta^{r_1+r_2} \mod p) \\ &= e_K(x_1 \cdot x_2, r_1 + r_2) \end{aligned}$$

2.4.1 ElGamal Problem:

This homomorphism is multiplicative. E-cash and e-voting would benefit from an additive homomorphism.

One solution: Modify ElGamal. Put the plaintext in the exponent. If we modify ElGamal so that

$$e_K(x, r) = (\alpha^r \mod p, \alpha^x \cdot \beta^r \mod p)$$

Then the homomorphism is

$$\begin{aligned} e_K(x_1, r_1) \cdot e_K(x_2, r_2) &= (\alpha^{r_1} \mod p, \alpha^{x_1} \cdot \beta^{r_1} \mod p) \cdot (\alpha^{r_2} \mod p, \alpha^{x_2} \cdot \beta^{r_2} \mod p) \\ &= (\alpha^{r_1} \cdot \alpha^{r_2} \mod p, \alpha^{x_1} \cdot \beta^{r_1} \cdot \alpha^{x_2} \cdot \beta^{r_2} \mod p) \\ &= (\alpha^{r_1+r_2} \mod p, \alpha x_1 + x_2 \cdot \beta^{r_1+r_2} \mod p) \\ &= e_K(x_1 + x_2, r_1 + r_2) \end{aligned}$$

The problem with this modification is that $d_K = \alpha^x$, introducing the discrete logarithm problem into the decryption. For large enough texts, this becomes impractical.

We would like another cryptosystem which takes advantage of this additive property of exponentiation but does so without extra decryption time.

Solution: the Paillier Cryptosystem.

2.5 Paillier Cryptosystem

- Introduced by Pascal Paillier in "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," Paillier 1999
- Probabilistic, asymmetric algorithm
- Based on the decisional composite residuosity assumption: Given composite n and integer z, it is hard to determine if y exists such that $\exists_? y : z \equiv y^n \pmod{n^2}$
- Homomorphic and self-blinding
- Extended by Damgard and Jurik in 2001 , modulo $n^2 \Rightarrow$ modulo n^{s+1}

2.5.1 Definition

Pick two large primes p and q and let $n = pq$. Let λ denote the Carmichael function, that is, $\lambda(n) = \mathrm{lcm}(p-1, q-1)$. Pick random $g \in \mathbb{Z}_n^*$ such that $L(g^\lambda \bmod n^2)$ is invertible modulo n (where $L(u) = u^{-1} \bmod n$). n and g are public; p and q (or λ) are private. For plaintext x and resulting ciphertext y, select a random $r \in \mathbb{Z}_n^*$. Then,

$$e_K(x, r) = g^x r^n \bmod n^2$$

$$d_K(y) = L\left(y^\lambda \bmod n^2\right) \cdot L\left(g^\lambda \bmod n^2\right)^{-1} \bmod n$$

$$e_K(x_1, r_1) \cdot e_K(x_2, r_2) = e_K(x_1 + x_2, r_1 + r_2)$$

2.5.2 Paillier Example: E-Voting

Suppose Soumik , Rajdeeep and Sandeep are running in an election. Each candidate is represented by a unique encrypted value. For instance, "01" represents Sandeep , "10 00" represents Rajdeep, and "10 00 00" represents Soumik. Only 6 people voted in the election, and the results are tabulated below:

Vote	Soumik	Rajdeeep	Sandeep	
1			✓	$\rightarrow$ 00 00 01 = 1
2		✓		$\rightarrow$ 00 01 00 = 4
3		✓		$\rightarrow$ 00 01 00 = 4
4			✓	$\rightarrow$ 00 00 01 = 1
5	✓			$\rightarrow$ 01 00 00 = 16
6			✓	$\rightarrow$ 00 00 01 = 1

Let $p = 5$ and $q = 7$. Then $n = 35$, $n^2 = 1225$, and $\lambda = 12$. g is chosen to be 141. For the first vote $x_1 = 1$, r is randomly chosen as 4. Then,

$$e_K(x_1, r_1) = e_K(1, 4) = 141^1 \cdot 4^{35} \mod 1225 = 141 \cdot 324 = 359 \mod 1225$$

All votes, r values, and resulting encryptions are shown below:

x	r	$e_K(x, r)$
1	4	359
4	17	173
4	26	486
1	12	1088
16	11	541
1	32	163

In order to sum the votes, we multiply the encrypted data modulo n^2:

$$359 \cdot 173 \cdot 486 \cdot 1088 \cdot 541 \cdot 163 \mod 1225 = 983$$

We then decrypt: $L(y^\lambda \bmod n^2) = L(983^{12} \bmod 1225) = 36^{-1} \bmod 35 = 1$

$$L(g^\lambda \bmod n^2) = L(141^{12} \bmod 1225) = 456^{-1} \bmod 35 = 13$$

$$d_K(y) = L(y^\lambda \bmod n^2) \cdot L(g^\lambda \bmod n^2)$$
$$= 1 \cdot 13^{-1} \bmod 35 = 27$$

We convert 27 to (01 02 03) for the final results. So, Sandeep is the winner.

2.6 Security Notions

Are homomorphic encryptions secure? Or are we losing anything in trade of these features?

Homomorphic encryption is malleable by design. A malleable cryptosystem is one in which anyone can intercept a ciphertext, transform it into another ciphertext, and then decrypt that into a plaintext that makes sense. Malleability is generally considered undesirable in a cryptosystem.

"Security" depends on the attack model and goal for rigorous cryptanalysis. In basic models of simple adversaries such as Cipher-only attack or Known-plaintext attack, it is as secure as normal encryption.

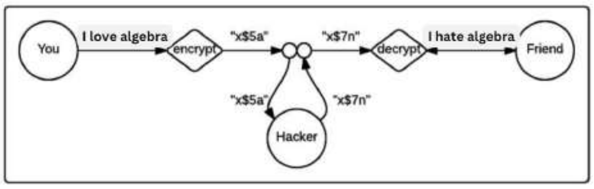

Figure 2.1: Malleability

Homomorphic systems should be malleable. Maybe you want to build a system that simply adds exclamation marks to whatever you send your friend. But you don't want the system to know what you're sending your friend; that's a secret. Malleable systems allow for multiple parties, especially in cloud-based environments, to operate on data without ever exposing it.

Chapter 3

Fully Homomorphic Encryption

Homomorphic in the name refers to homomorphism in algebra: the encryption and decryption functions can be thought of as homomorphisms between plaintext and ciphertext spaces. Encryption with an additional evaluation capability for computing over encrypted data without access to the secret key. The result of such a computation remains encrypted. It includes multiple types of encryption schemes that can perform different classes of computations with different capabilities.

Scheme Type	**Capability and Description**
Partially homomorphic encryption	Supports evaluation of circuits consisting of only one type of gate, e.g., addition or multiplication.
Somewhat homomorphic encryption	Can evaluate multiple types of gates, but only for a subset of circuits.
Fully homomorphic encryption (FHE)	Allows evaluation of arbitrary circuits composed of multiple types of gates of unbounded depth and is the strongest notion of homomorphic encryption.

- Up until above, the homomorphic systems described have been partially homomorphic (PHE). They preserve the structures of multiplication or division, but cannot do both
- The additive and multiplicative preservation of a Ring Homomorphism modulo 2 directly correspond to the XOR and AND operations of a circuit

- Applications:
  - Private queries on search engines - The search engine would be able to return encrypted data with out every decrypting the query
  - Cloud Computing - Storing encrypted data on the cloud is seemingly useless; no manipulation of the data can be obtained with out allowing the cloud access and/or decrypting the data off the cloud

3.1 Fully Homorphic Encryption

Fully homomorphic systems are homomorphic systems in which any kind of mathematical operation can be performed on the cipher text. Fully homomorphic systems do exist today, and their optimizations since 2009 have made them practical for some applications. Craig Gentry was the first to suggest that they could be theoretically possible. He was able to create a system that was homomorphic in two ways, and those two ways allowed full homomorphism.

3.1.1 Craig Gentry

Centers around a function which introduces a certain level of noise into the encryption. Each operation on the ciphertext results in compounding noise, resolved with the bootstrapability of the encryption. Each re-encryption cuts down the noise. Analogy to Alice's jewelry shop. Involves operations on Ideal Lattices. Allows for less complex circuit implementation, corresponding to the structure of Rings.Gentry 2010

Gentry uses the analogy of a jewelry shop owner in his thesis to describe why fully homomorphic systems should be, and are, possible. Imagine that Alice is a jewelery store owner. She has employees that assemble products from raw materials like diamonds and gold. But, she's worried about the possibility of theft. So, she designs boxes that have gloves attached to them. Employees can stick their hands into the box to assemble the products, but they cannot take anything out of the box because only Alice has the key. So, Alice's employees can do operations on the secure data (the jewelry) without ever having the possibility of taking that secure data out.

3.1.2 Challanges

Gentry's system incorporates an amount of noise into the cryptographic process. Each successive encryption introduces more noise into the system, which is why Gentry's initial design is impractical (though it was later improved upon). It is impractical to use noise because eventually the system needs to be restarted because the added noise makes the entire system much slower. This system relies on ideal Lattice Based Cryptography to simplify much of the system's design. Mattsson 2021

However, the combination of the noise production followed by the noise reduction makes the scheme completely impractical. Complexity grows as more and more operations are performed (inherent limitation of the algorithm). Gentry has stated that in order to perform one search on Google using this encryption, the amount of computations needed would increase by a trillion. More schemes have been introduced to try and decrease this complexity, but all rely on the same principles. Despite this impracticality, Gentry's discovery is an amazing breakthrough in cryptography and proves that (at least theoretical) fully homomorphic encryption schemes exist.

Conclusion

In conclusion, homomorphic encryption represents a remarkable advancement in the field of cryptography, offering the tantalizing prospect of performing computations on encrypted data without compromising privacy. While challenges such as noise accumulation and computational complexity persist, the theoretical underpinnings and practical applications of homomorphic encryption continue to drive innovation and research in the field.

The existence of fully homomorphic encryption schemes, as demonstrated by pioneering work such as that of Craig Gentry, underscores the boundless possibilities and enduring relevance of cryptography in safeguarding privacy and enabling secure computation in an increasingly interconnected world. As we look to the future, further advancements in homomorphic encryption techniques, alongside complementary developments in areas such as post-quantum cryptography and secure multi-party computation, hold promise for addressing the remaining limitations and expanding the practical utility of this transformative technology. It is an open area of research, with ongoing efforts focused on achieving fully homomorphic encryption, where computations of arbitrary complexity can be performed

In the pursuit of a more secure and privacy-preserving digital society, homomorphic encryption stands as a beacon of innovation and hope, offering a path forward towards realizing the vision of secure computation over encrypted data.

References

Boneh, Dan and Shoup, Victor (Jan. 2023). *A graduate course in applied cryptography.* Draft 0.6. Stanford.

Gentry, Craig (2010). “Computing Arbitrary Functions of Encrypted Data”. In: *Association for Computing Machinery.*

Mattsson, Ulf (June 2021). *Fully HE-based (FHE) processing remains 1,000 to 1,000,000 times slower than equivalent plaintext operations.* Protegrity Corp.

Paillier, Pascal (1999). “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *EUROCRYPT.*

Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). “On data banks and privacy homomorphisms”. In: *Foundations of Secure Computation.*

Appendix A

Number Theory

A.1 Euler's Totient Function

In number theory, Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n. It is written using the Greek letter phi as $\varphi(n)$ or $\phi(n)$, and may also be called Euler's phi function. In other words, it is the number of integers k in the range $1 \leq k \leq n$ for which the greatest common divisor $\gcd(n, k)$ is equal to 1. The integers k of this form are sometimes referred to as totatives of n.

For example, the totatives of $n = 9$ are the six numbers $1, 2, 4, 5, 7$ and 8. They are all relatively prime to 9, but the other three numbers in this range, $3, 6$, and 9 are not, since $\gcd(9, 3) = \gcd(9, 6) = 3$ and $\gcd(9, 9) = 9$. Therefore, $\phi(9) = 6$. As another example, $\phi(1) = 1$ since for $n = 1$ the only integer in the range from 1 to n is 1 itself, and $\gcd(1, 1) = 1$.

Euler's totient function is a multiplicative function, meaning that if two numbers m and n are relatively prime, then $\phi(mn) = \phi(m)\phi(n)$. This function gives the order of the multiplicative group of integers modulo n (the group of units of the ring $\mathbb{Z}/n\mathbb{Z}$). It is also used for defining the RSA encryption system.

A.2 Discrete Logarithm

In mathematics, for given real numbers a and b, the logarithm $\log_b a$ is a number x such that $b^x = a$. Analogously, in any group G, powers b^k can be defined for all integers k, and the discrete logarithm $\log_b a$ is an integer k such that $b^k = a$. In number theory, the more commonly used term is index: we can write $x = \text{ind}_r a \pmod{m}$ (read "the index of a to the base r modulo m") for $r^x \equiv a \pmod{m}$ if r is a primitive root of m and $\gcd(a, m) = 1$.

Discrete logarithms are quickly computable in a few special cases; however, no efficient method is known for computing them in general.

Definition: Let G be any group. Denote its group operation by multiplication and its identity element by 1. Let b be any element of G. For any positive integer k, the expression b^k denotes the product of b with itself k times. Similarly, let b^{-k} denote the product of b^{-1} with itself k times. For $k = 0$, the kth power is the identity: $b^0 = 1$. Let a also be an element of G. An integer k that solves the equation $b^k = a$ is termed a discrete logarithm (or simply logarithm, in this context) of a to the base b. One writes $k = \log_b a$.

A.3 Public-Key Cryptography

Public-Private keys, they are used in asymmetric cryptography. In these systems each key pair consists of a public key known to all, and a corresponding private key which is secret to the reciever. Key pairs are generated with key generation algorithms based on mathematical problems termed one-way functions. Security of public-key cryptography depends on keeping the private key secret; the public key can be openly distributed without compromising security.

In a public-key encryption system, anyone with a public key can encrypt a message, yielding a ciphertext, but only those who know the corresponding private key can decrypt the ciphertext to obtain the original message.

For example, a journalist can publish the public key of an encryption key pair on a web site so that sources can send secret messages to the news organization in ciphertext. Only the journalist who knows the corresponding private key can decrypt the ciphertexts to obtain the sources' messages—an eavesdropper reading email on its way to the journalist cannot decrypt the ciphertexts. However, public-key encryption does not conceal metadata like what computer a source used to send a message, when they sent it, or how long it is. Public-key encryption on its own also does not tell the recipient anything about who sent a message—it just conceals the content of a message in a ciphertext that can only be decrypted with the private key.

In a digital signature system, a sender can use a private key together with a message to create a signature. Anyone with the corresponding public key can verify whether the signature matches the message, but a forger who does not know the private key cannot find any message/signature pair that will pass verification with the public key.