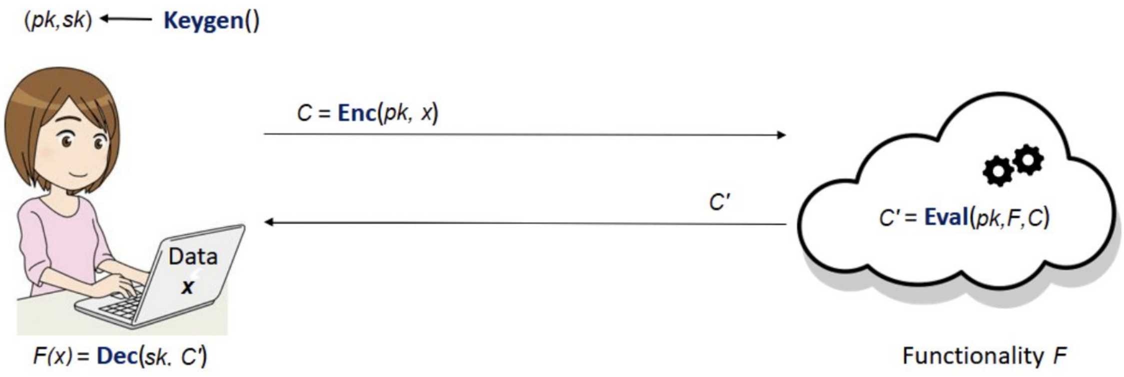
Homomorphic Arithmetic Operations on Ciphertexts

Really Special Amigos (RSA)

Motivation / Example use case

Homomorphic encryption allows the ability to analyze and gather information on an encrypted dataset without ever having to decrypt the data.

- Alice has medical data of patients at her hospital and wants to keep this private
- There is a company x which provides high quality predictive analysis which can help Alice treat her patients better
- Alice sends the encrypted data (C) to X. X uses the public key, C, and some function F to compute the analysis or predictions (C')
- Alice then uses her private/secret key to decrypt the output of X's function to get the actionable information



https://bit-ml.github.io/blog/post/homomorphic-encryption-toy-implementation-in-python/

Somewhat Homomorphic (SHE) vs Fully Homomorphic (FHE)

With each operation on encrypted data, the amount of noise in the result grows. Given enough operations, the noise will make accurate decryption impossible.

Somewhat Homomorphic: a number of addition/multiplication operations decided beforehand to minimize total noise

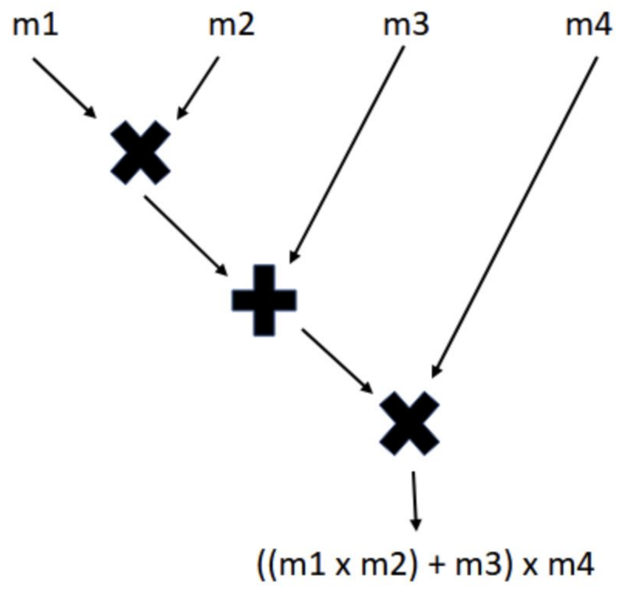
Transitioning to FHE: Gentry’s scheme and “bootstrapping” SHE schemes

- Computationally very expensive

FHE: the number of operations is unbounded.

Partially Homomorphic Encryption (PHE)

$$F(m_1, m_2, m_3, m_4) = m_1 \times m_2 \times m_4 + m_3 \times m_4.$$



Main difference between PHE and FHE: PHE usually only supports one type of operation: Addition **or** Multiplication

- The decrypted result of addition or multiplication of two ciphertexts is the same as the result of the same operation over the plaintexts.
- Encrypt(a) + Encrypt(b) = Encrypt(a + b)

All complex operations can be broken down into multiple addition and multiplication operations (AND, OR, and NOT boolean gates)

Solutions Considered

State of the Art (FHE) Libraries

- TFHE [TFHE/Google - C++]
- SEAL [Microsoft - C++]
- HElib [IBM - C++/Python]
- Concrete [Zama - Rust/Python]

Other Crypto Libraries

- bit-ml scheme [Python]

Selected

- Java Homomorphic Encryption
  - Specifically the Paillier cryptosystem components
  - Client-Server model

Paillier’s Scheme

Partially homomorphic scheme, relies on the Decisional Composite Residuosity Assumption:

Given non-prime integer n and integer z, it is hard to determine whether there exists y such that $z \equiv y^n \pmod{n^2}$

Developed in 1999 by Pascal Paillier (now the CTO at Zama)

Chosen because it is relatively simple to implement compared to FHE solutions

Paillier’s Scheme - Continued

Supported Operations

- Addition:
    - Ciphertext + Plaintext
    - Ciphertext + Ciphertext
- Multiplication:
    - Ciphertext x Plaintext
    - ~~Ciphertext x Ciphertext~~
- Comparison:
    - Ciphertext > Ciphertext

Pascal Paillier

Breakdown of Project

- Alice.java & Bob.java
  - Instances of the client and the server. Holds private keys and public keys
- security.paillier
  - Public-private key pair generations
  - Encryption with public key
  - Description with private key
  - Addition & multiplication using public key
  - Signature and verification

Challenges

- We sometimes witnessed noise in our final result which manifested as a +/- 1 discrepancy between the encrypted answer and plaintext answer
- Division is challenging in this HE scheme
  - For x/y, x mod y must equal 0 or division will return an incorrect result.
    - Averages
    - Median (for even set sizes)

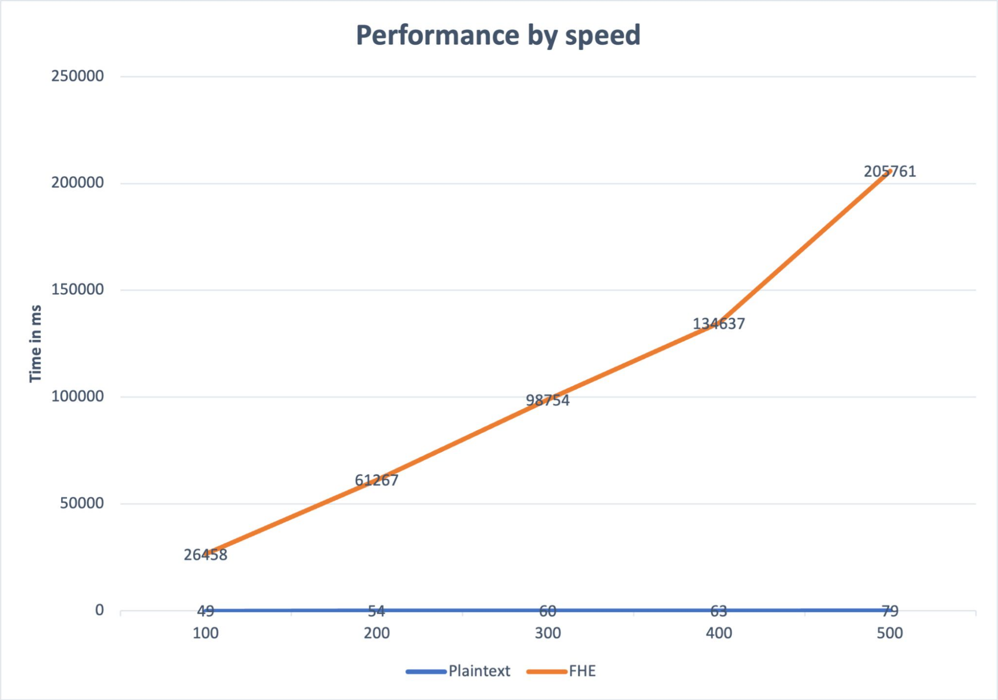
Dataset

- Due to the nature of medical data, cybersecurity in healthcare has become a unique challenge
  - Some primary concerns for healthcare facilities:
    - Man-in-the-middle (MITM) attacks
    - Attacks to network vulnerabilities
-

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	1	67	0	1	1	Private	Urban	229	37	formerly smoked	1
1	31112	1	80	0	1	1	Private	Rural	106	33	never smoked	1
2	60182	0	49	0	0	1	Private	Urban	171	34	smokes	1
3	1665	0	79	1	0	1	Self-employed	Rural	174	24	never smoked	1
4	56669	1	81	0	0	1	Private	Urban	186	29	formerly smoked	1
...	...	...	...	...	...	...	...	...	...	...	...	...
4883	14180	0	13	0	0	0	children	Rural	103	19	Unknown	0
4884	44873	0	81	0	0	1	Self-employed	Urban	125	40	never smoked	0
4885	19723	0	35	0	0	1	Self-employed	Rural	83	31	never smoked	0
4886	37544	1	51	0	0	1	Private	Rural	166	26	formerly smoked	0
4887	44679	0	44	0	0	1	Govt_job	Urban	85	26	Unknown	0

4888 rows × 12 columns

Comparison of protocols



- Plaintext algorithm:
  - Quadratic
  - Scalable but not privacy preserving

- FHE:
  - Not quadratic
  - Poor performance: slow computation speed, accuracy problems
  - commercially infeasible for computationally-heavy applications

Demo Video