Client-optimized Algorithms and Acceleration for Encrypted Compute Offloading
Part 1

Matias Mazzanti

28 of March de 2023

Schedule

- What is HE?
- What is FHE?
- Schemes
- Math notation

What is HE?

Homomorphic Encrpytion (HE) $\rightarrow$ form of encryption. Earlys 80s

Operate **directly** with encrypted data (without the need of decrypt).

Typical use case:

- Client encrypted his data with his own secret key.
- Send this and the public key to the server.
- Server operates with out decrypting (e.g. Machine Learning).
- Server send back the result in the encryption form.
- Client desencripts the result with the secret key.

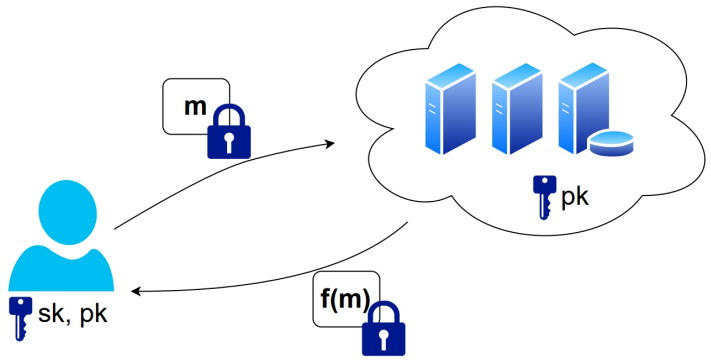


Figure: Paper:https://ia.cr/2022/657

Potentially very useful for cloud computing use.

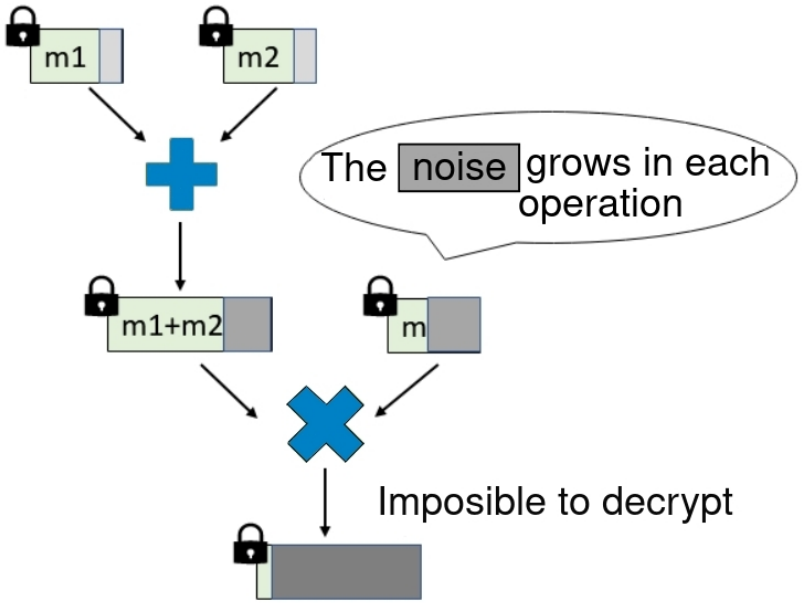
Present: Orders of magnitude **slower** for real use.

FHE

In general, HE schemes use `Ring Learning With Errors` (RLWE) that adds some sort of error to the encryption.

This error grows in each operation (particulary wiht multiplications).

If the error is too big the decrpytion will not work.



Fully Homomorphic Encrpytion (FHE): Unlimited number of operations. In this type of schemes it use `Bootstrapping` that are techniques that refresh this error.

Bootstrapping

Bootstrapping vs leveling

o para la siguiente clase?

Schemes

Many schemes. **Warning**: Many acronyms!!!

Popular schemes by types:

- Operations on integers: BFV and BGV.
- Operations on real numbers: CKKS.
- Operations on boolean gates: FHEW and THFHE.

The parameters of all schemes are application-specific and desire security.

Math notation

Most schemes works in a Polynomial Ring domain.

N will be the degree of the Polynomials (N-1).

q and t will be the mod of the coefficients.

Ring: a set with addition, substraction and multiplication. (and other propertys, comutative, associativity, etc).

This operations of elements in a ring return elements in the ring $\rightarrow$ take modulus.

Math notation

BFV, BGV and CKKS works with: $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$

$\mathbb{Z}[X]$ = Set of Polynomials with integer coffiencients.

$\mathbb{Z}_q[X]$ = Set of Polynomials with integer coffiencients mod q.

$\mathbb{Z}_q[X]/(X^N + 1)$ = The Polynomials degree N-1.

BFV Primitives

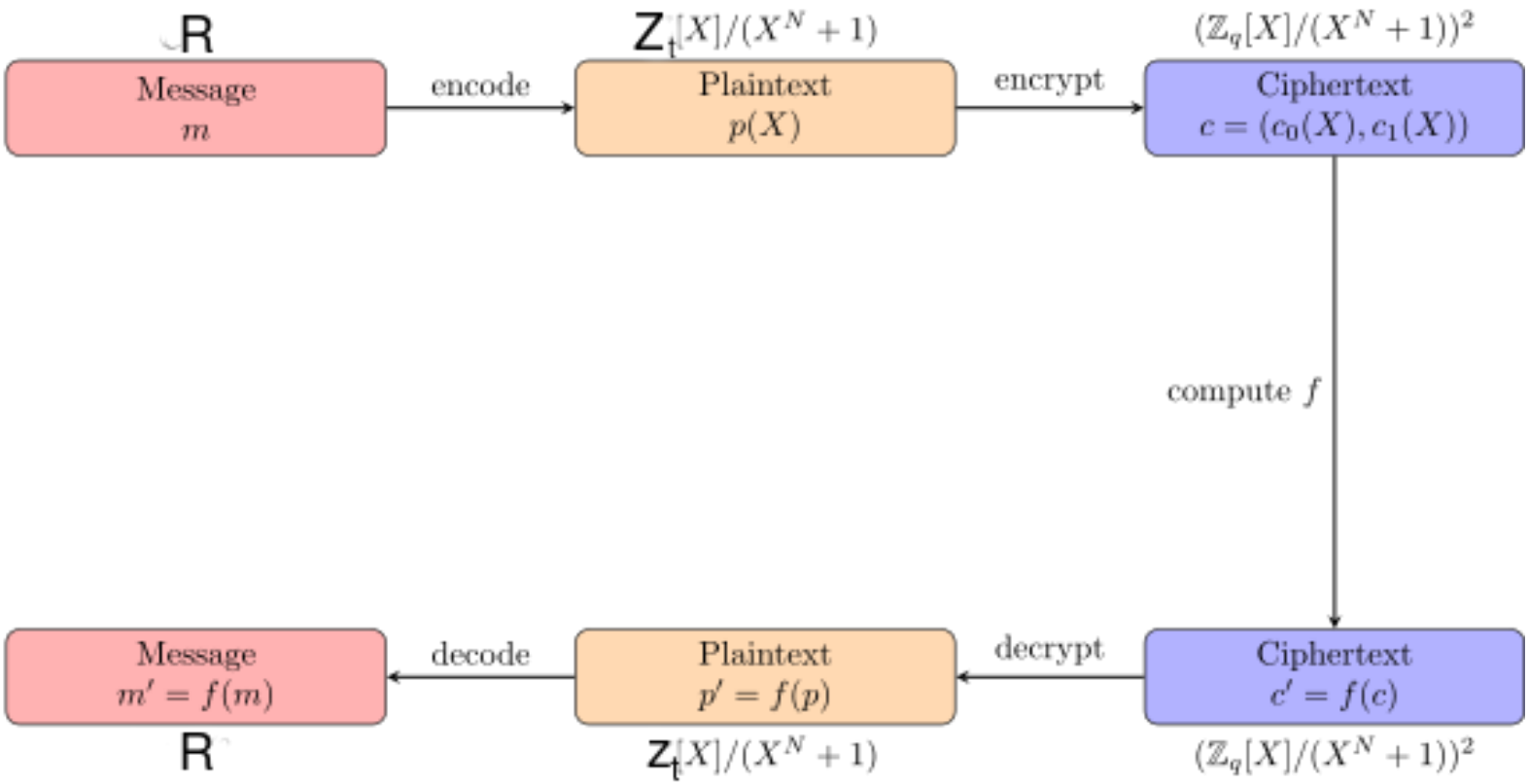
Basic Primitives (simplified):

- ParamGen(λ)$\rightarrow$ Params. Security $\lambda \sim 2^{\lambda}$ operation to decrypt with prob 1.
- KeyGen(Params) $\rightarrow$ SK, PK, EvalK
- Encrypt(PK, m) $\rightarrow$ c
- Decrypt(SK, c) $\rightarrow$ m
- EvalAdd(c_1, c_2) $\rightarrow$ c_3
- EvalMult(c_1, c_2) $\rightarrow$ c_3
- Relinerize(c_3, EvalK) $\rightarrow$ c_3'

Workflow

Encryption has two phases:

- From an integer m to Plaintext
- From plain text to ciphertext c.



BFV

The parameters depends in the security level needed and amount of operations that will made.

The parameters depends in the security level needed and amount of operations that will made.

The parameters depends in the security level needed and amount of operations that will made.

Some usual parameters to get the sense of what we are doing:

N = 8192, 16384, 32768.

q = 2^{218}, 2^{438}, 2^{881}.

t = less than q.

Implementations: Residue Number System (RNS) (stay with 64bits arithmetics).

Encryption

Encryption has two phases:

- From an integer m to Plaintext M $\rightarrow$ Easy, binary representation as Polynomial coefficient.

- From plaintext to ciphertext using $PK = (PK_1, PK_2)$:

  - Sample a polynomail u from $\mathcal{R}_2$.
  - Sample polynomial errors e_1 and e_2 form a Gaussian distribution.
  - Calculate $\triangle = \lfloor q/t \rfloor$
  - Encryption of plaintext M is $C = (C_1, C_2)$

$$C_1 = [PK_1 * u + e_1 + \triangle M]_q$$
$$C_2 = [PK_2 * u + e_2]_q$$

Multiplication

Multiplication: computation cost and error grow.

$$C_{mul} = C * C' = (C_1 * C'_1, C_1 * C'_2 + C_2 * C'_1, C_2 * C'_2) = (C_a, C_b, C_c)$$

Problem: If we keep multipling, the number of Polynomials will grow and will be needed the squared of the SK.

Relinerize: $EvalK = (-PK_1 + SK^2, PK_2)$

$$C_{mul} = (C_a, C_b, C_c) \approx ([C_a + EvalK_1 * C_c]_q, [C_b + EvalK_2 * C_c]_q)$$

Other problem: we are multiplying the errors of the two ciphertexts, the result error grows a lot!

Solution:

- Select the lowest parametrs for te exact numbers of operations needed (HE).
- Use Bootstrapping us needed (FHE). Even more demanding (multiple multiplications like C_{mul}).

Recap

Rembember! All things are high degree polynomials with huge coeffient.

For a single multiplication of two numbers:

- Enconde and encrypt: Sample the errors and parts of the Keys and multiply them to generate the Keys.
- Multiplication: Multply the different parts of the ciphertexts.
- Relinerize: Multiply the result with the evaluation key.

Each homomorphic operation is $10^3 \sim 10^6$ times slower than unencrypted.

Large footprint in memory. An encrypted date can occupy 10^5 more space.

The schemes are iterative, operating many time with each data.

Research

- Accelerating the polynomial multiplication: For these they use Number Theoretic Transform (NTT), a Discreate Fourier Transform (DFT) over a ring.
- New schemes.
- New upgrades in the schemes (like RNS).
- New methods of Bootstrapping.
- Software Optimization.
- Hardware Acceleration (GPUs, FPGA, specific designs).

New approach

The usual way of approaching this problem of speed (and others): accelerating the server part.

The new approach: accelerate the client side.

CHACO: **C**lient-aided **H**E for **O**paque **C**ompute **O**ffloading.

- Another way to operate by using multiple times the client.
- New algorithm: rotational redundancy.
- Reduction of parameters values.
- Minimice client costs.
- Hardware acceleration for client: CHACO-TACO

FHE requirements and problems: