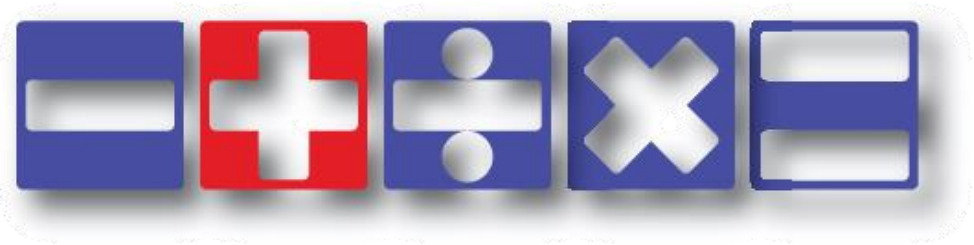
Arithmetic questions in Fully Homomorphic Encryption

Anamaria Costache

ARITH 2025



NTNU

Introduction

Fully Homomorphic Encryption

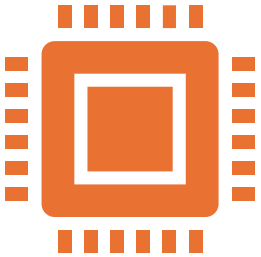
- Allows to compute on encrypted data in a way that matches the computation on the underlying plaintext

$$f\big(Enc(m)\big) = Enc(f(m))$$

- Proposed as a concept in ‘78, first construction in ‘09

On data banks and privacy homomorphisms. RL Rivest, L Adleman, ML Dertouzos. Foundations of Secure Computation 1978.
Fully homomorphic encryption using ideal lattices. Craig Gentry. ACM symposium on Theory of computing, 2009

Applications



Countless applications, most easily illustrated with *outsourcing computation*

Can use FHE to construct many complex privacy-preserving protocols

Secure voting

Private Set Intersection

Private Information Retrieval

Secure medical computations/ analytics

Secure biometric identification/ verification

…

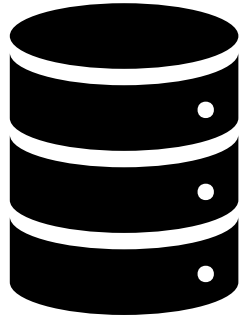
$(sk, pk) \leftarrow KeyGen(1^{\lambda})$

$ct \leftarrow Enc(m, pk)$

$ct' \leftarrow f(ct, pk) = Enc(f(m), pk)$

(ct, f, pk)

ct'



$f(m) \leftarrow Dec(ct', sk)$

Privacy-preserving Machine Learning

FHE is particularly well-suited to Privacy-Preserving Machine Learning applications

Enables a client (making the query) to keep their data private

Can also enable the server to **hide their model**

Many challenges here: more on this later!

Terminology

Partial homomorphic encryption

Can support **one of** additions or multiplications

Somewhat homomorphic encryption

Can support both, but a **limited** number

Fully homomorphic encryption

Unlimited number of operations

Technical Introduction

Learning with Errors

Definition (Learning with Errors): Let q be a positive integer, let n be a dimension and an error distribution χ. For a secret $s \in \mathbb{Z}_q^n$, the LWE distribution outputs pairs of the form

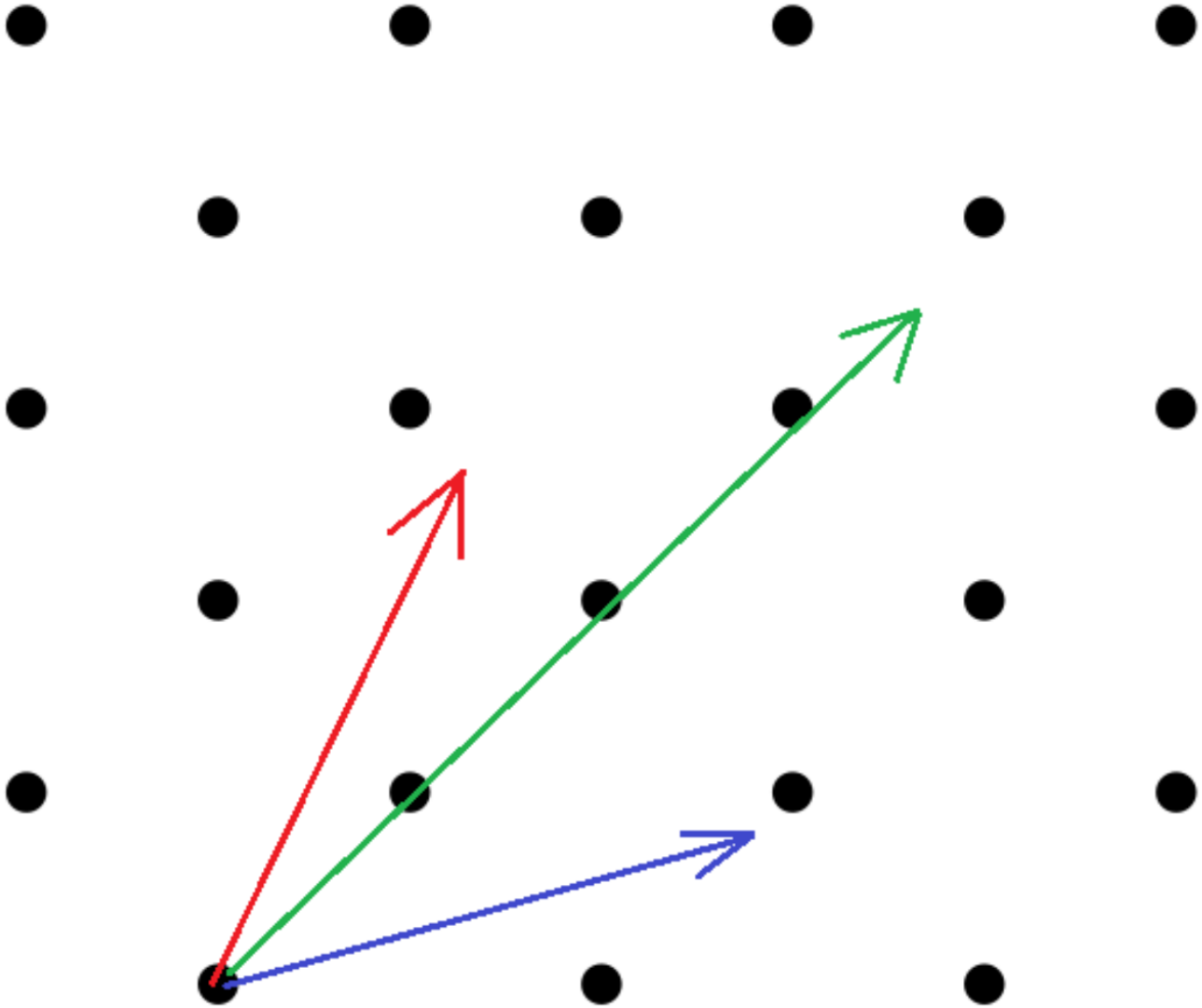
$$(a_i, b_i) = (a_i, b_i = < a_i, s > + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q,$$

where e_i are sampled according to χ and $a_i \in \mathbb{Z}_q^n$ are sampled uniformly at random,.

- The LWE **decision** problem is: given pairs of the form (a_i, b_i), decide if they were sampled according to the LWE distribution or uniformly at random.
- The LWE **search** problem is: given pairs (a_i, b_i), recover the secret s.
- We will use the *Ring-Learning with Errors* version of this problem, where $\mathbb{Z}_q^n$ is replaced by a polynomial ring

On lattices, learning with errors, random linear codes and cryptography. Oded Regev. Journal of the ACM 2009.

Noise in FHE



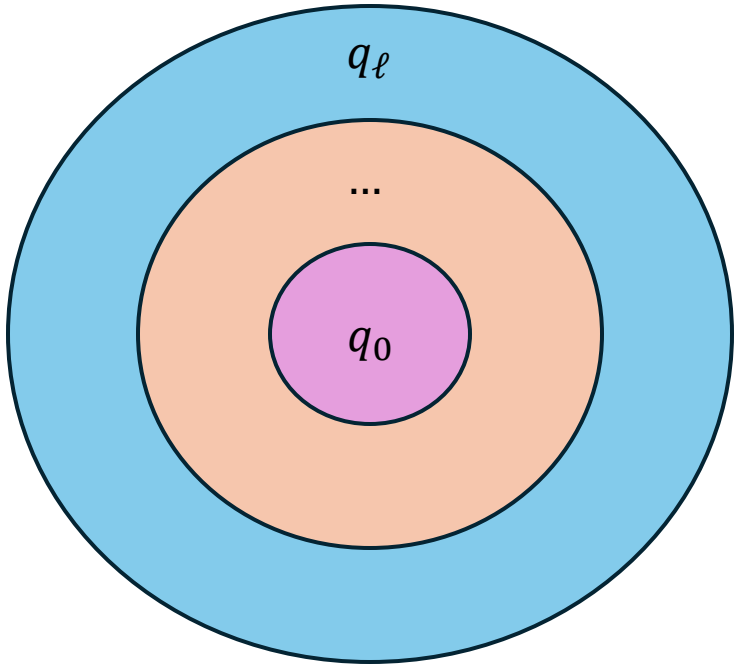
Two paradigms for handling noise

Bootstrapping

- Decrypting removes **all** noise
- But that would reveal the message
- Perform decryption function homomorphically, reduce the noise

Levelled schemes

- "Layer" the ciphertext space into levels
- After each multiplication, divide and round
- Controls the noise growth



Four generations of schemes

First generation

- Gentry's original scheme (2009)

Second generation

- BGV, BFV, (NTRU)
- Good packing capabilities, slow *bootstrapping*, exact computations, additions and multiplications

Third generation

- GSW, FHEW, TFHE
- Very fast *bootstrapping*, non-linear functions, low packing capabilities

Fourth generation

- CKKS
- (very) Good packing capabilities, slow *bootstrapping,* additions and multiplications

Parameters (for this talk)

- We often operate on the space $\mathcal{R}_q = \mathbb{Z}[x]/(\phi_n, q)$ where ϕ_n is the $2n$-th cyclotomic polynomial

$$\phi_n(x) = x^n + 1,$$

for $n = 2^k$, for some k, and q is the (Ring-)LWE modulus.

- Ciphertexts are of the form $(c_0, c_1) \in \mathcal{R}_q^2$.
- Typically, q is chosen to be a product of ℓ primes $p_0, \dots, p_{\ell-1}$
- We write $q_k = \prod_{i=0}^{k} p_i$
- Each prime $p_i, i \in \{0, \dots, \ell - 1\}$ is roughly of the same size, and is chosen such that $p_i \equiv 1 \ (mod\ 2n)$
- In particular, we can use the Chinese Remainder Theorem (CRT) to write the following isomorphism

$$\mathcal{R}_{q_{\ell-1}} \cong \frac{\mathbb{Z}[x]}{\phi_n, p_0} \times \cdots \times \frac{\mathbb{Z}[x]}{\phi_n, p_{\ell-1}}.$$

- We are particularly interested in using the **Residue Number System (RNS)**

Encodings

Mod q and mod t operations **do not commute!** The noise typically needs to remain **smaller than q**!

A ciphertext is a pair (c_0, c_1), under some secret key s.

	Lower bits	Upper bits	Approximate
Message encoding	$m + t \cdot e$	$\Delta \cdot m + e$	$m + e$
Decryption	$\left[[c_0 + sc_1]_q\right]_t$	$\left[\frac{t}{q} \lceil [c_0 + sc_1]_q \rfloor\right]_t$	$[c_0 + sc_1]_q$

Δ is a scaling factor, t is the plaintext modulus, q is the ciphertext modulus

$[\cdot]_q$ denotes modular reduction (centered around 0) mod q, $|\cdot|_q$, "regular" modular reduction

BFV operations - deep dive

BFV Encryption

- Let t be a plaintext modulus, let q_ℓ be as before. Let $e, e_0, e_1 \leftarrow \chi$ be noise terms sampled from the noise distribution χ, and let $s \leftarrow \mathcal{S}$ be a secret key
- Then, the public key is

$$(p_0, p_1) = pk = [-a \cdot s + e, a]_q$$

- We encrypt a message $m \in \mathcal{R}_t$ as follows. Let $v \leftarrow \mathcal{S}$ be an ephemeral key

$$Enc(m, pk) = [\Delta m + vp_0 + e_0, vp_1 + e_1]_q = (c_0, c_1) \in \mathcal{R}_q^2$$

- **Addition** is simple (linear)
  - Coordinate-wise
  - Noise growth is very small

$[\cdot]_q$ denotes modular reduction (centered around 0) mod q, $|\cdot|_q$, "regular" modular reduction

Multiplication – the challenges

Multiplication is a more complex operation

A “naïve” approach (simply multiplying the components) leads to unmanageable noise growth, leading to losing the message

Use **base decomposition instead**

Multiplication I

- **Multiplication** is a more complex operation, and consists of **two** steps. Multiplying the "naïve" way would result in unmanageable noise growth.

- Consider two ciphertexts, encrypted under the same key

$$ct = Enc(m_0, pk) = [\Delta m_0 + v\, p_0 + e_0, v p_1 + e_1]_q = (c_0, c_1) \in \mathcal{R}_q^2$$
$$ct' = Enc(m_1, pk) = [\Delta m_1 + v'\, p_0 + e'_0, v' p_1 + e'_1]_q = (c'_0, c'_1) \in \mathcal{R}_q^2$$

  - **First step:**

$$(d_0, d_1, d_2) = (c_0 \cdot c'_0, c_0 \cdot c'_1 + c'_0 \cdot c_1, c_1 \cdot c'_1)$$

$$\left(\left\lfloor \frac{t \cdot d_0}{q} \right\rceil, \left\lfloor \frac{t \cdot d_1}{q} \right\rceil, \left\lfloor \frac{t \cdot d_2}{q} \right\rceil \right)$$

  - This results in a 3-element ciphertext that decrypts under $\boldsymbol{s^2}$, rather than s

$$d_0 + s \cdot d_1 + s^2 \cdot d_2 = \Delta m + E\ ,$$

for some noise E.

Multiplication II

- We have a 3-element ciphertext that decrypts under $\boldsymbol{s^2}$, rather than s

$$d_0 + s \cdot d_1 + s^2 \cdot d_2 = \Delta m + E\ ,$$

for some noise E.

- We want to recover a 2-element ciphertext, that decrypts under s. A simple solution:

$$(d_0 + s^2 d_2, d_1) \text{ decrypts as } (d_0 + s^2 \cdot d_2) + s \cdot d_1 = \Delta m + E.$$

- **But:** s^2 is secret! So, we encrypt it as $(a'' \cdot s + e' + s^2, -a'')$.

- Now, we can form the ciphertext

$$(d_0 + d_2 \cdot (a'' \cdot s + e'' + s^2), d_1 - d_2 \cdot a'')$$

- Can decrypt as

$$d_0 + d_2 \cdot a'' \cdot s + d_2 \cdot e'' + d_2 \cdot s^2 + s \cdot d_1 - s \cdot d_2 \cdot a''$$
$$= d_0 + s \cdot d_1 + s^2 \cdot d_2 + d_2 \cdot e'' = \Delta m + E + d_2 \cdot e''$$

Remember decryption is $c_0 + s \cdot c_1$!

Remember the server computes this and s^2 is secret!

Multiplication III

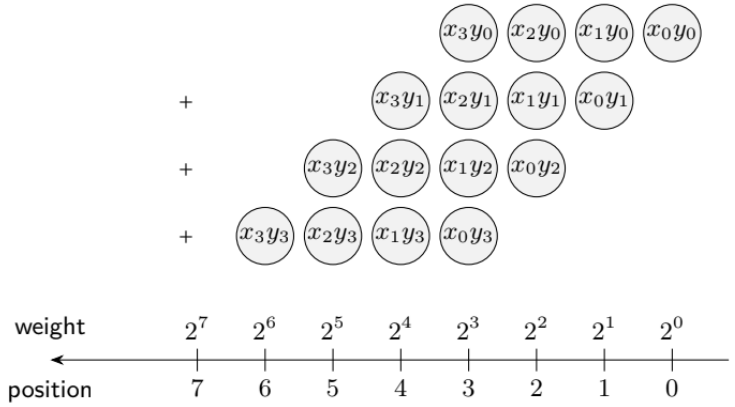
- Introduce the two following functions, which will help **manage the noise growth**

$$\forall a \in \mathcal{R} = \begin{cases} \mathcal{D}_{\omega,q}(a) & = \left([a]_\omega, [\lfloor a\omega^{-1} \rfloor]_\omega, \ldots, [\lfloor a\omega^{-\kappa} \rfloor]_\omega\right) \in \mathcal{R}_\omega^\kappa \\ \mathcal{P}_{\omega,q}(a) & = \left([a]_q, [a\omega]_q, \ldots, [a\omega^\kappa]_q\right) \in \mathcal{R}_q^\kappa \end{cases}$$

$$\forall a, b \in \mathcal{R}, \langle \mathcal{D}_{\omega,q}(a), \mathcal{P}_{\omega,q}(b) \rangle \equiv ab \pmod{q}$$

- Also need a **relinearization key** as follows (i.e. an encryption of s^2)

$$rlk = \left([\mathcal{P}_{\omega,q}(s^2) - (e + sa)]_q, a\right)$$



- Finally, we compute the output ciphertext as

$$ct_{\text{out}} = \left([d_0 + \langle \mathcal{D}_{\omega,q}(d_2), rlk_0 \rangle]_q, [d_1 + \langle \mathcal{D}_{\omega,q}(d_2), rlk_1 \rangle]_q\right)$$

Credit to Florent de Dinechin and Martin Kumm for the figure!

Challenges for RNS

The product in the first step is very **large** – would normally require a lift to $\mathbb{Z}$

Dividing and rounding is incompatible with RNS (only exact division)

Decomposition in base ω requires a positional system

(A) Solution for Decryption

- Recall that division is performed as

$$\left\lfloor \frac{t}{q} [c_0 + s \cdot c_1]_q \right\rceil$$

- Exact divisions are possible in RNS!

$$\left\lfloor \frac{t}{q} \mid c_0 + s \cdot c_1 \mid_q \right\rfloor = \frac{t \mid c_0 + s \cdot c_1 \mid_q - \mid t \cdot c_0 + s \cdot c_1 \mid_q}{q}$$

- This results in an **approximate** decryption

- Then, correct the approximation

A Full RNS Variant of FV-like Somewhat Homomorphic Encryption Schemes. Jean-Claude Bajard, Julien Eynard, Anward Hasan, Vincent Zucca. SAC 2016.

(A) Solution for Multiplication

- Introduce an auxiliary base to contain the product in step 1 (which is large!) Replaces the lift to $\mathbb{Z}$

$$FastBaseConv = \left(\sum_{i=1}^{k} | \, x_i \frac{q_i}{q} \, |_{q_i} \times \frac{q}{q_i} \pmod{m} \right)_{m \in \mathcal{B}}$$

- Use RNS instead – if ω has the same size as the moduli in q, this is a good approximation

$$\xi_q(\bar{c_2}) = \left(\, | \, \bar{c_2} \frac{q_1}{q} \, |_{q_1}, \ldots, | \, \bar{c_2} \frac{q_k}{q} \, |_{q_k} \, \right)$$

$$\mathcal{P}_{RNS,q} = \left(\, | \, s^2 \frac{q}{q_1} \, |_q, \ldots, | \, s^2 \frac{q}{q_k} \, |_q \, \right)$$

A Full RNS Variant of FV-like Somewhat Homomorphic Encryption Schemes. Jean-Claude Bajard, Julien Eynard, Anward Hasan, Vincent Zucca. SAC 2016.

Decomposition base 2^k

- Elements in the ciphertext space are polynomials $a \in \mathcal{R}_q$
- Write $a = \sum_{i=0}^{N-1} a_i X^i$
- Introduce a second variable Y
- Write further $a_i = \sum_{j=0}^{L-1} a_{i,j} 2^{jk} = \sum_{j=0}^{L-1} a_{i,j} Y^j$, where $L = D/k$, for some chosen precision D
- This can replace the RNS representation

Revisiting Key Decomposition Techniques in FHE: Simpler, Fast and More Generic. Mariya Georgieva Belorgey, Sergiu Carpov, Nicolas Gama, Sandra Guasch, Dimitar Jetchev. AsiaCrypt 2024

Accelerating HE Operations from Key Decomposition Technique. Miran Kim, Dongwon Lee, Jinyeong Seo, Yongsoo Song. Crypto 2023

An HPR variant of the FV scheme: Computationally Cheaper, Asymptotically Faster. Jean-Claude Bajard, Julien Eynard, Paulo Martins, Leonel Sousa, Vincent Zucca. Transactions on Computers 2019.

Plaintext types

Plaintext types

Plaintext types for the different schemes

- (small) Integers for TFHE
- Polynomials in $\mathcal{R}_t = \frac{\mathbb{Z}[x]}{x^N+1,t}$ for BGV, BFV
- (fixed precision) vectors in $\mathbb{C}^{N/2}$ for CKKS (N is the degree of the ring)

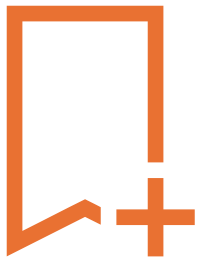
One interesting optimisation question

- BGV, BFV, CKKS can only evaluate **additions, multiplications, rotations**
- This means that non-linear functions cannot be computed **exactly** (ReLu, SoftMax etc)
- How to best approximate these?
    - Remember, multiplications are expensive!

This is different for TFHE!

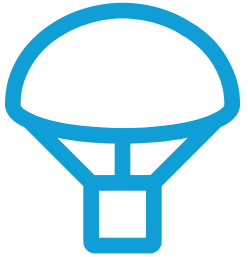
- Can evaluate lookup tables efficiently! So can evaluate most nonlinear functions **exactly** and **efficiently**
- However, it performs best on messages of 5 bits of precision
- Cannot pack as many values as the other schemes

Data types



Integers (polynomials) are the most natural to deal with

Fixed point requires some care

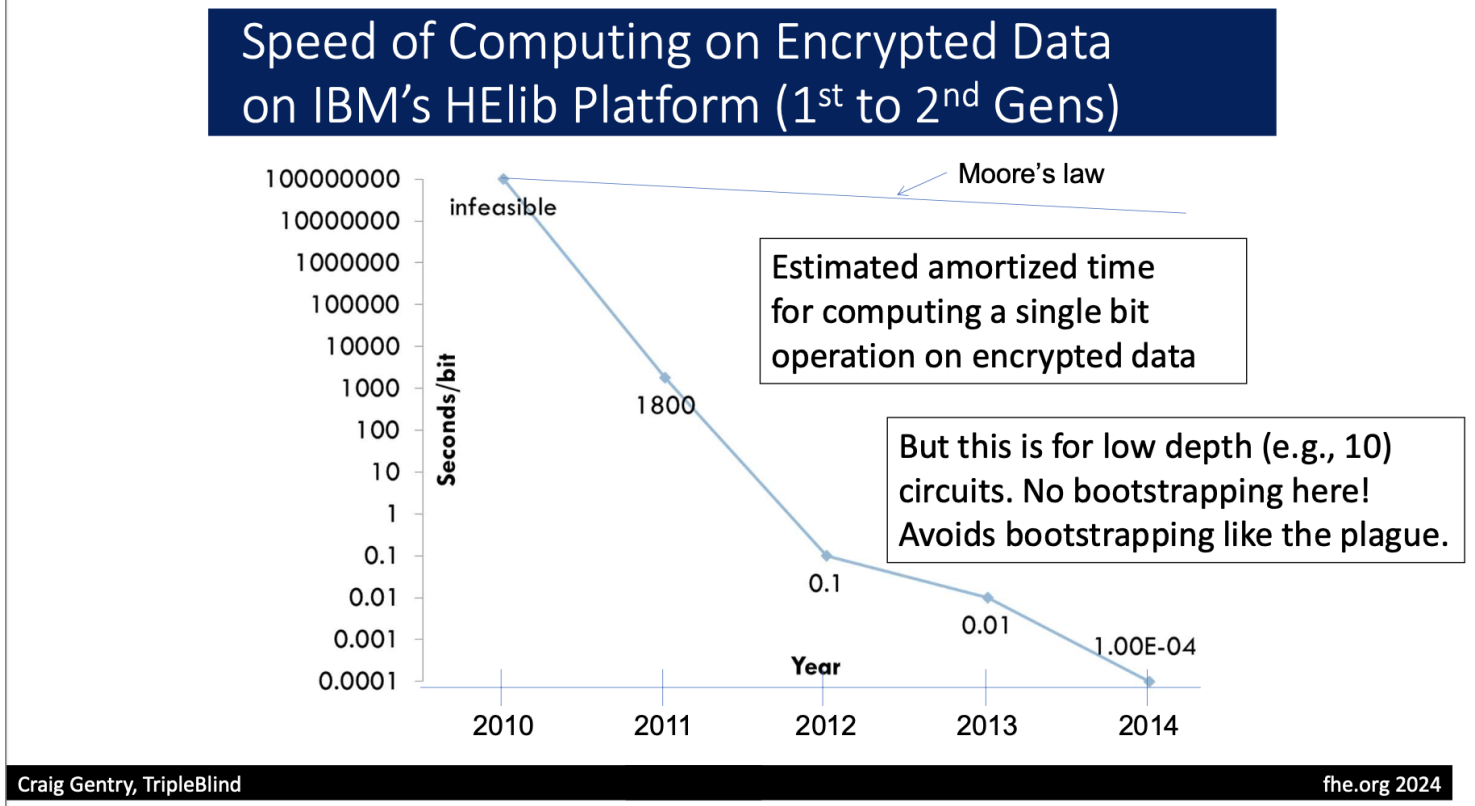


Floating point is more delicate

An Efficient Encrypted Floating-Point Representation Using HEAAN and TFHE, Subin Moon, Younho Lee. SCN 2020.
Simple Encrypted Arithmetic Library, Kim Laine. 2017
Fixed-point Arithmetic in SHE schemes. *Anamaria Costache*, Nigel P. Smart, Srinivas Vivek and Alexander Waller. SAC 2016.
TFHE gets real: an Efficient and Flexible Homomorphic Floating-Point Arithmetic. Loris Bergerat, Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila and Samuel Tap. CHES 2025.

Conclusion and open questions

Evolution of speed of computing



Slide from *FHE: Past, Present and Future*. Craig Gentry, fhe.org meetup 2024

Cryptographic-specific challenges

Verifiable computation:

FHE does not, by itself, offer any guarantees about the *correctness of the* function evaluated

Must operate in an *honest-but-curious* setting, where we trust that the server runs the operation intended, correctly

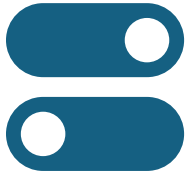
Verifiable Computation (VC) addresses this question exactly – but many efficiency issues

Circuit privacy:

FHE does not, by itself, hide the circuit to be evaluated!

Particularly relevant in Privacy-Preserving Machine Learning (PPML) settings, where server will not want to release the model

Some recent work, but much remains to be done



Scheme switching:

Some schemes have good packing capabilities, some have very fast operations and can handle nonlinear operations

Would be good to have an efficient way of efficiently switching between different types of schemes

Thank you for your attention!

Special thanks to Laurent Imbert for helpful advice in preparing this talk!

- Gentry 2009: **Fully Homomorphic Encryption from Ideal Lattices.** Craig Gentry. ACM Symposium on Theory of Computing 2009.
- BGV: **(Levelled) Fully Homomorphic Encryption without Bootstrapping.** Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan. ACM Transactions on Computation Theory 2012.
- BFV:
  - **Fully Homomorphic Encryption without Modulus Switching from GapSVP**. Zvika Brakerski**.** Crypto 2012.
  - **Somewhat Practical Fully Homomorphic Encryption.** Junfeng Fan, Frederik Vercauteren. ePrint archive 2012.
- GSW: **Homomorphic Encryption from Learning with Errors: Conceptually Simpler, Asymptotically-Faster, Attribute-Based**. Craig Gentry, Amit Sahai, Brent Waters. Crypto 2013.
- FHEW: **FHEW: Bootstrapping Homomorphic Encryption in Less than a Second**. Léo Ducas, Daniele Micciancio. Eurocrypt 2014.
- TFHE: **Fast Fully Homomorphic Encryption over the Torus.** Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène. Journal of Cryptography 2020.
- CKKS: **Homomorphic Encryption for Arithmetic of Approximate Numbers.** Jung Hee Cheon, Andrey Kim, Miran Kim, Yonsoo Song. Asiacrypt 2017.