Challenges and open problems in Fully Homomorphic Encryption

Anamaria Costache

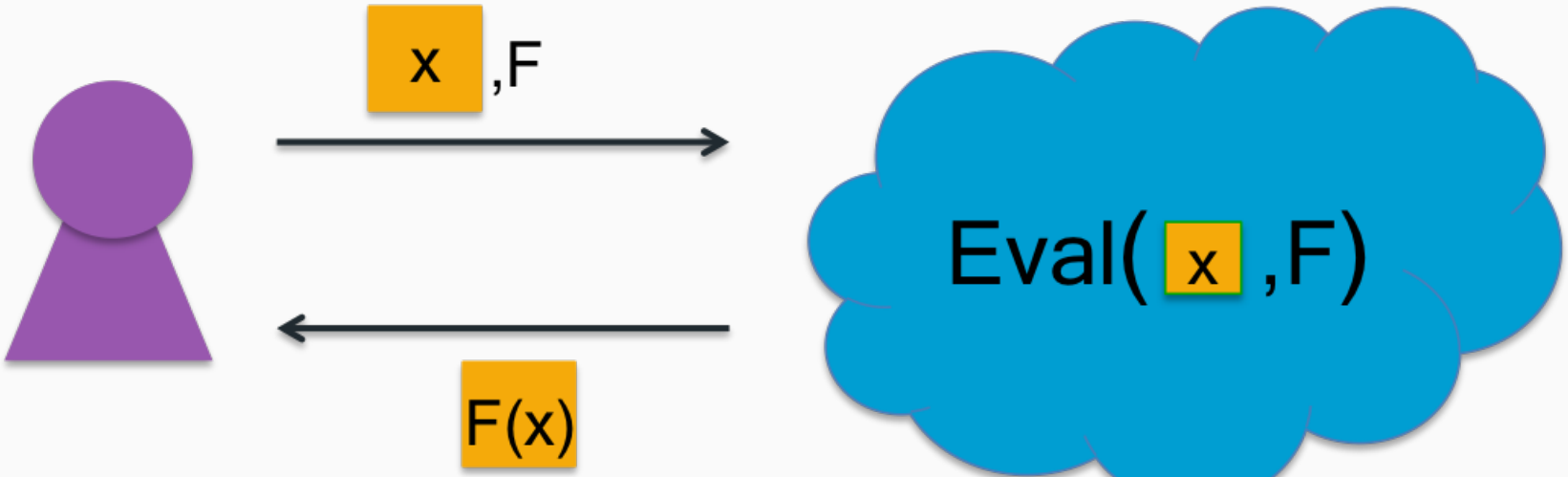
27 July 2022

Norwegian University of
Science and Technology

Table of contents

Introduction, history and applications

What is Homomorphic Encryption?



x Encryption of x

F(x) Encryption of F(x)

Thanks to Rachel Player for this slide!

Fully Homomorphic Encryption: a brief history

1978	Proposed as open problem [RAD78]
2009	Gentry's blueprint [Gentry09]
2010	First Implementation
2010	[DGHV10]
2012	Second generation schemes: BGV [BGV12], B/FV [Bra12, FV12], ...
2013	Third generation schemes: GSW [GSW13], ...
	First open source libraries available
2014	FHEW [DM14]
2016	TFHE introduces fast bootstrapping (less than 0.1 seconds!) [CGGI16]
2017	Standardisation effort begins
	Approximate homomorphic encryption: CKKS [CKKS17]
2020	ISO efforts begin

Community efforts

Actively maintained libraries

Name	Implements	Available at
HElib	BGV, CKKS	`github.com/homenc/HElib`
SEAL	BFV, CKKS	`github.com/Microsoft/SEAL`
Palisade	BFV, BGV and more!	`git.njit.edu/palisade/PALISADE`
Concrete	TFHE	`github.com/zama-ai/concrete`
Lattigo	CKKS, BFV	`github.com/tuneinsight/lattigo`

Standardisation effort[1]

- Homomorphic Encryption Standard recommends secure parameters
- White papers on applications, APIs, schemes etc.
- Usability focus: use-cases
- Standardisation: ISO/ ISE

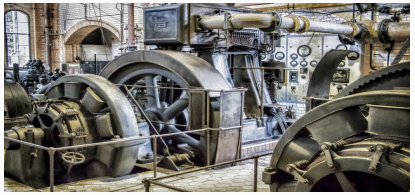
[1] `homomorphicencryption.org`, `fhe.org`

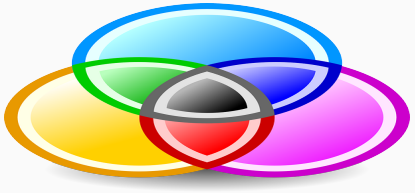
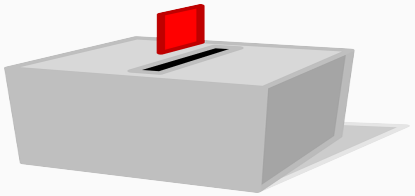
Many possible applications

- Medical
- Genomics
- Machine Learning
- Statistics
- Smart cities
- Cyber physical systems
- Private information retrieval
- Database search
- Private set intersection
- Electronic voting
- ...





Schemes

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$
  - $Q = q_{L-1} = \prod_{j=0}^{L-1} p_j$ the product all the primes is the "top" modulus

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$
  - $Q = q_{L-1} = \prod_{j=0}^{L-1} p_j$ the product all the primes is the "top" modulus
- Gaussian error distribution χ with standard deviation σ

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$
  - $Q = q_{L-1} = \prod_{j=0}^{L-1} p_j$ the product all the primes is the "top" modulus
- Gaussian error distribution χ with standard deviation σ
- Secret distribution S

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$
  - $Q = q_{L-1} = \prod_{j=0}^{L-1} p_j$ the product all the primes is the "top" modulus
- Gaussian error distribution χ with standard deviation σ
- Secret distribution S
- Plaintext modulus t[1]

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$
  - $Q = q_{L-1} = \prod_{j=0}^{L-1} p_j$ the product all the primes is the "top" modulus
- Gaussian error distribution χ with standard deviation σ
- Secret distribution S
- Plaintext modulus t[1]
- Security parameter λ

[1] For BGV and BFV

A Ring-LWE-based (levelled) FHE scheme is parameterised by:

- Dimension n (here always a power of two)
- Power-of-two cyclotomic polynomial $\Phi_n = x^n + 1$
- Polynomial ring $R_q = \mathbb{Z}[x]/(\Phi_n, q)$
- Ciphertext moduli q_k
  - There are L primes $p_0, \ldots, p_{L-1}$
  - The chain of moduli is formed as $q_k = \prod_{j=0}^{k} p_j$
  - $Q = q_{L-1} = \prod_{j=0}^{L-1} p_j$ the product all the primes is the "top" modulus
- Gaussian error distribution χ with standard deviation σ
- Secret distribution S
- Plaintext modulus t[1]
- Security parameter λ
- $\mathbb{Z}_q$ means $\mathbb{Z}/q\mathbb{Z}$, $[\cdot]_q$ means reduction modulo q

[1] For BGV and BFV

The CKKS scheme - basic operations

Ciphertexts are elements $(c_0, c_1) \in R_q^2$.

$\texttt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\texttt{sk} = (1, s)$.

The CKKS scheme - basic operations

Ciphertexts are elements $(c_0, c_1) \in R_q^2$.

$\texttt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\texttt{sk} = (1, s)$.

$\texttt{PublicKeyGen}(\texttt{sk})$: Recall $\texttt{sk} = (1, s)$ and sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi$. Output $\texttt{pk} = ([-as + e]_q, a)$.

The CKKS scheme - basic operations

Ciphertexts are elements $(\mathtt{c}_0, \mathtt{c}_1) \in R_q^2$.

$\mathtt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\mathtt{sk} = (1, s)$.

$\mathtt{PublicKeyGen}(\mathtt{sk})$: Recall $\mathtt{sk} = (1, s)$ and sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi$. Output $\mathtt{pk} = ([-as + e]_q, a)$.

$\mathtt{Encrypt}(\mathtt{pk}, m)$: For the message $m \in R$. Let $\mathtt{pk} = (p_0, p_1)$, sample $v \leftarrow S$ and $e_1, e_2 \leftarrow \chi$. Output $\mathtt{ct} = ([m + e_1 + p_0 v]_q, [p_1 v + e_2]_q)$.

The CKKS scheme - basic operations

Ciphertexts are elements $(\mathrm{c}_0, \mathrm{c}_1) \in R_q^2$.

$\texttt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\texttt{sk} = (1, s)$.

$\texttt{PublicKeyGen}(\texttt{sk})$: Recall $\texttt{sk} = (1, s)$ and sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi$. Output $\texttt{pk} = ([-as + e]_q, a)$.

$\texttt{Encrypt}(\texttt{pk}, m)$: For the message $m \in R$. Let $\texttt{pk} = (p_0, p_1)$, sample $v \leftarrow S$ and $e_1, e_2 \leftarrow \chi$. Output $\texttt{ct} = ([m + e_1 + p_0 v]_q, [p_1 v + e_2]_q)$.

$\texttt{Decrypt}(\texttt{sk}, \texttt{ct})$: Let $\texttt{ct} = (c_0, c_1)$. Output $m' = [c_0 + c_1 s]_q$.

The CKKS scheme - homomorphic operations

- Addition is coordinate-wise
- Multiplication, at a high level, corresponds to three steps:
  - `Pre-Multiply`: from two ciphertexts in R_q^2 that both decrypt under $\mathtt{sk}$, we have one ciphertext in R_q^3 that decrypts under $\mathtt{sk}^2$

The CKKS scheme - homomorphic operations

- Addition is coordinate-wise
- Multiplication, at a high level, corresponds to three steps:
  - `Pre-Multiply`: from two ciphertexts in R_q^2 that both decrypt under `sk`, we have one ciphertext in R_q^3 that decrypts under $\texttt{sk}^2$
  - `Relin`: From one ciphertext in R_q^3 that decrypts under `sk`, we go to one ciphertext in R_q^2 that decrypts under $\texttt{sk}^2$

The CKKS scheme - homomorphic operations

- Addition is coordinate-wise
- Multiplication, at a high level, corresponds to three steps:
  - `Pre-Multiply`: from two ciphertexts in R_q^2 that both decrypt under $\mathtt{sk}$, we have one ciphertext in R_q^3 that decrypts under $\mathtt{sk}^2$
  - `Relin`: From one ciphertext in R_q^3 that decrypts under $\mathtt{sk}$, we go to one ciphertext in R_q^2 that decrypts under $\mathtt{sk}^2$
  - `Rescale`: Multiply by q'/q and round to reduce the noise and output one ciphertext in $R_{q'}$ that decrypts under $\mathtt{sk}$

The CKKS scheme - homomorphic operations

We have $\mathtt{ct}_0 = (\mathtt{ct}_0[0], \mathtt{ct}_0[1])$ and $\mathtt{ct}_1 = (\mathtt{ct}_1[0], \mathtt{ct}_1[1])$ (mod q).

$\mathtt{Add}(\mathtt{ct}_0, \mathtt{ct}_1)$: Output $\mathtt{ct} = ([\mathtt{ct}_0[0] + \mathtt{ct}_1[0]]_q, [\mathtt{ct}_0[1] + \mathtt{ct}_1[1]]_q)$.

The CKKS scheme - homomorphic operations

We have $\mathtt{ct}_0 = (\mathtt{ct}_0[0], \mathtt{ct}_0[1])$ and $\mathtt{ct}_1 = (\mathtt{ct}_1[0], \mathtt{ct}_1[1])$ (mod q).

$\mathtt{Add}(\mathtt{ct}_0, \mathtt{ct}_1)$: Output $\mathtt{ct} = ([\mathtt{ct}_0[0] + \mathtt{ct}_1[0]]_q, [\mathtt{ct}_0[1] + \mathtt{ct}_1[1]]_q)$.

$\mathtt{Pre\text{-}Multiply}(\mathtt{ct}_0, \mathtt{ct}_1)$: Set $d_0 = [\mathtt{ct}_0[0]\mathtt{ct}_1[0]]_q$,
$d_1 = [\mathtt{ct}_0[0]\mathtt{ct}_1[1] + \mathtt{ct}_0[1]\mathtt{ct}_1[0]]_q$, and
$d_2 = [\mathtt{ct}_0[1]\mathtt{ct}_1[1]]_q$. Output $\mathtt{ct} = (d_0, d_1, d_2)$.

The CKKS scheme - homomorphic operations

We have $\mathtt{ct}_0 = (\mathtt{ct}_0[0], \mathtt{ct}_0[1])$ and $\mathtt{ct}_1 = (\mathtt{ct}_1[0], \mathtt{ct}_1[1])$ (mod q).

$\mathtt{Add}(\mathtt{ct}_0, \mathtt{ct}_1)$: Output $\mathtt{ct} = ([\mathtt{ct}_0[0] + \mathtt{ct}_1[0]]_q, [\mathtt{ct}_0[1] + \mathtt{ct}_1[1]]_q)$.

$\mathtt{Pre\text{-}Multiply}(\mathtt{ct}_0, \mathtt{ct}_1)$: Set $d_0 = [\mathtt{ct}_0[0]\mathtt{ct}_1[0]]_q$,
$d_1 = [\mathtt{ct}_0[0]\mathtt{ct}_1[1] + \mathtt{ct}_0[1]\mathtt{ct}_1[0]]_q$, and
$d_2 = [\mathtt{ct}_0[1]\mathtt{ct}_1[1]]_q$. Output $\mathtt{ct} = (d_0, d_1, d_2)$.

$\mathtt{KeySwitch}(\mathtt{ct}, \mathtt{evk})$: Let $\mathtt{ct}[0] = d_0$, $\mathtt{ct}[1] = d_1$ and $\mathtt{ct}[2] = d_2$. Recall
$\mathtt{evk}[0] = -a's + e' + Ps^2$ and $\mathtt{evk}[1] = a'$. Set
$c_0' = [d_0 + \lfloor P^{-1} \cdot d_2 \cdot (-a's + e' + Ps^2) \rceil]_q$, and
$c_1' = [d_1 + \lfloor P^{-1} \cdot d_2 \cdot a' \rceil]_q$. Output $\mathtt{ct}' = (c_0', c_1')$.

The CKKS scheme - homomorphic operations

We have $\mathtt{ct}_0 = (\mathtt{ct}_0[0], \mathtt{ct}_0[1])$ and $\mathtt{ct}_1 = (\mathtt{ct}_1[0], \mathtt{ct}_1[1])$ (mod q).

$\mathtt{Add}(\mathtt{ct}_0, \mathtt{ct}_1)$: Output $\mathtt{ct} = ([\mathtt{ct}_0[0] + \mathtt{ct}_1[0]]_q, [\mathtt{ct}_0[1] + \mathtt{ct}_1[1]]_q)$.

$\mathtt{Pre\text{-}Multiply}(\mathtt{ct}_0, \mathtt{ct}_1)$: Set $d_0 = [\mathtt{ct}_0[0]\mathtt{ct}_1[0]]_q$,
$d_1 = [\mathtt{ct}_0[0]\mathtt{ct}_1[1] + \mathtt{ct}_0[1]\mathtt{ct}_1[0]]_q$, and
$d_2 = [\mathtt{ct}_0[1]\mathtt{ct}_1[1]]_q$. Output $\mathtt{ct} = (d_0, d_1, d_2)$.

$\mathtt{KeySwitch}(\mathtt{ct}, \mathtt{evk})$: Let $\mathtt{ct}[0] = d_0$, $\mathtt{ct}[1] = d_1$ and $\mathtt{ct}[2] = d_2$. Recall
$\mathtt{evk}[0] = -a's + e' + Ps^2$ and $\mathtt{evk}[1] = a'$. Set
$c_0' = [d_0 + \lfloor P^{-1} \cdot d_2 \cdot (-a's + e' + Ps^2) \rceil]_q$, and
$c_1' = [d_1 + \lfloor P^{-1} \cdot d_2 \cdot a' \rceil]_q$. Output $\mathtt{ct}' = (c_0', c_1')$.

$\mathtt{Rescale}(\mathtt{ct}, q, q')$: Let $\mathtt{ct} = (c_0, c_1)$. Set $c_0' = \left[\left\lfloor \frac{q' c_0}{q} \right\rceil\right]_{q'}$ and
$c_1' = \left[\left\lfloor \frac{q' c_1}{q} \right\rceil\right]_{q'}$. Output $\mathtt{ct} = (c_0', c_1')$.

The BGV scheme

$\texttt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\texttt{sk} = s$.

$\texttt{PublicKeyGen}(\texttt{sk})$: Set $s = \texttt{sk}$ and sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi$. Output $\texttt{pk} = ([-(as + et)]_q, a)$.

$\texttt{Encrypt}(\texttt{pk}, m)$: For the message $m \in R_t$. Let $\texttt{pk} = (p_0, p_1)$, sample $u \leftarrow S$ and $e_1, e_2 \leftarrow \chi$. Output $\texttt{ct} = ([m + p_0 u + te_1]_q, [p_1 u + te_2]_q)$.

$\texttt{Decrypt}(\texttt{sk}, \texttt{ct})$: Let $s = \texttt{sk}$ and $\texttt{ct} = (c_0, c_1)$. Output $m' = [[c_0 + c_1 s]_q]_t$.

The BFV scheme

$\texttt{SecretKeyGen}(\lambda)$: Sample $s \leftarrow S$ and output $\texttt{sk} = s$.

$\texttt{PublicKeyGen}(\texttt{sk})$: Set $s = \texttt{sk}$ and sample $a \leftarrow R_q$ uniformly at random and $e \leftarrow \chi$. Output $\texttt{pk} = ([-(as + e)]_q, a)$.

$\texttt{Encrypt}(\texttt{pk}, m)$: For the message $m \in R_t$. Let $\texttt{pk} = (p_0, p_1)$, sample $u \leftarrow S$ and $e_1, e_2 \leftarrow \chi$. Output $\texttt{ct} = ([\Delta m + p_0 u + e_1]_q, [p_1 u + e_2]_q)$.

$\texttt{Decrypt}(\texttt{sk}, \texttt{ct})$: Let $s = \texttt{sk}$ and $\texttt{ct} = (c_0, c_1)$. Output $m' = \left[\left\lfloor \frac{t}{q} [c_0 + c_1 s]_q \right\rceil \right]_t$.

A side-by-side comparison

Scheme	BGV	BFV	CKKS
Message encoding	$m + t \cdot e$	$\Delta \cdot m + e$	$m + e$
Message encoding	Lower bits	Upper bits	Approximate encryption
Decryption	$m' = \left[[c_0 + c_1 s]_q\right]_t$	$m' = \left[\left\lfloor \frac{t}{q}[c_0 + c_1 s]_q \right\rceil\right]_t$	$m' = [c_0 + c_1 s]_q$
Multiplication	$m_0 m_1 + t^2 e_0 e_1 + t(e_0 m_1 + e_1 m_0)$	$\Delta^2 m_0 m_1 + \Delta(e_0 m_1 + e_1 m_0) + e_0 e_1$	$m_0 m_1 + m_1 e_0 + m_0 e_1 + e_0 e_1$

Noise growth is much slower in CKKS.

Packing

Encoding and Decoding for CKKS

The CKKS scheme uses the canonical embedding to define an encoding from the message space $\mathbb{C}^{N/2}$ to the "plaintext" space $\mathbb{Z}[X]/(X^N+1)$ in the following way: an isomorphism $\tau : \mathbb{R}[X]/(X^N+1) \to \mathbb{C}^{N/2}$ can be defined via considering the canonical embedding restricted to $N/2$ of the $2N^{\text{th}}$ primitive roots and discarding conjugates. Encoding and decoding then use this map τ, as well as a precision parameter Δ, as follows:

$$\text{Encode}(\mathbf{z}, \Delta) = \left\lfloor \Delta \tau^{-1}(\mathbf{z}) \right\rceil, \qquad \text{Decode}(m, \Delta) = \frac{1}{\Delta}\tau(m),$$

where $\mathbf{z} \in \mathbb{C}^{N/2}$, $m \in \mathbb{Z}_q[X]/(X^N+1)$ and $\lfloor \cdot \rceil$ is taken coefficient-wise.

Can pack up to $N/2$ values into a single ciphertext.

Plaintext packing in BGV - CRT for rings

Theorem
Let R be a ring, and $I_0, \ldots, I_{k-1}$ be pairwise coprime ideals. Let $I = \bigcap_{i=0}^{k-1} I_i$. Then we have that

$$R/I \cong R/I_0 \times \ldots \times R/I_{k-1}.$$

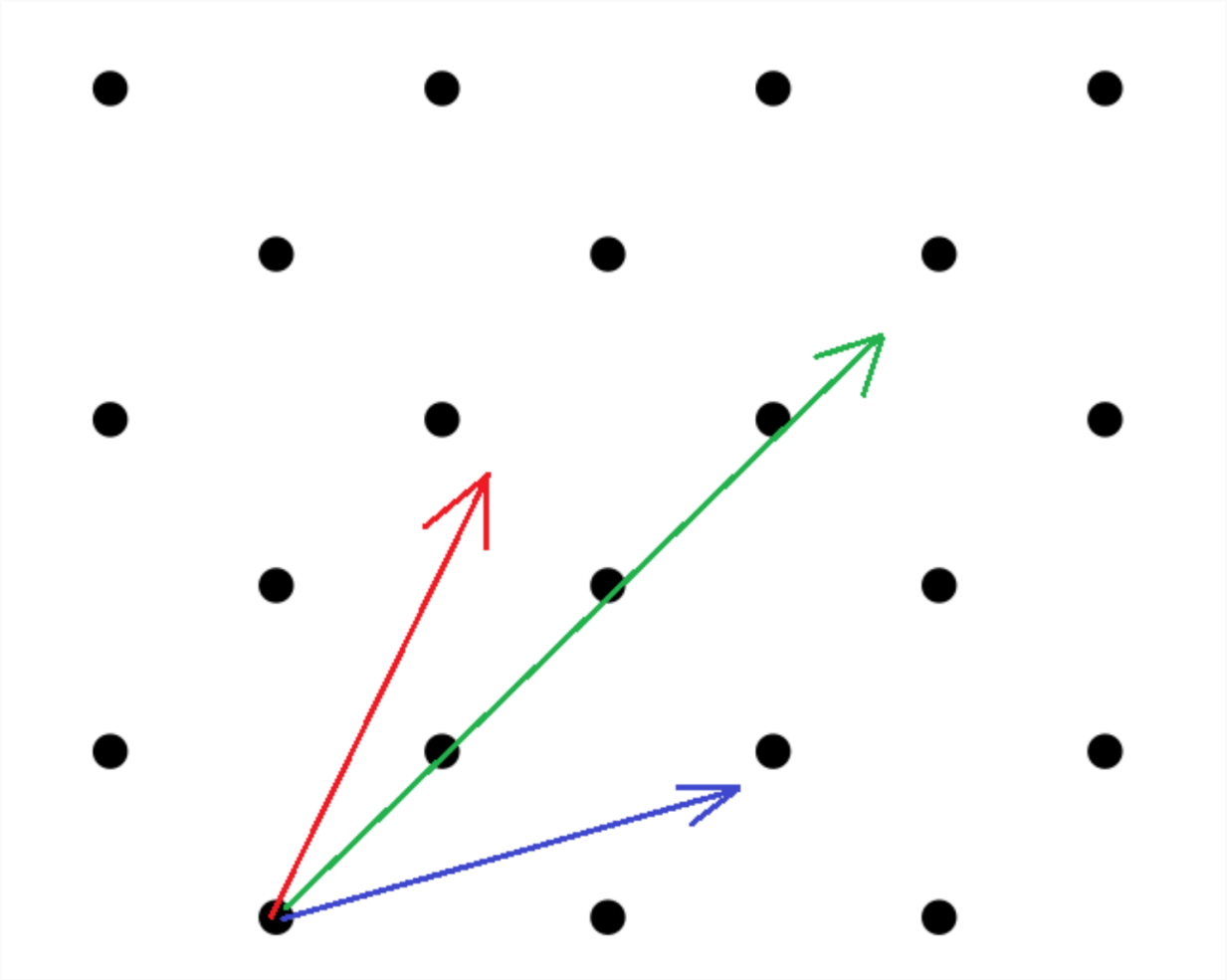
Now let $R = \mathbb{Z}_q[x]/(\Phi_n)$, where Φ_n is the nth cyclotomic polynomial, q is a ciphertext modulus and $\mathbb{Z}_t[x]/(\Phi_n)$ is the plaintext space. Consider $\mathbb{Z}/t\mathbb{Z}$ such that it contains a primitve nth root of unity. Then, Φ_n factors into ℓ irreducible factors

$$\Phi_n = \prod_{i=0}^{\ell-1} f_i(x) \pmod{t}.$$

The ideals (f_i) are pairwise coprime in R, and we can apply the Chinese Remainder Theorem for rings to obtain plaintext slots.

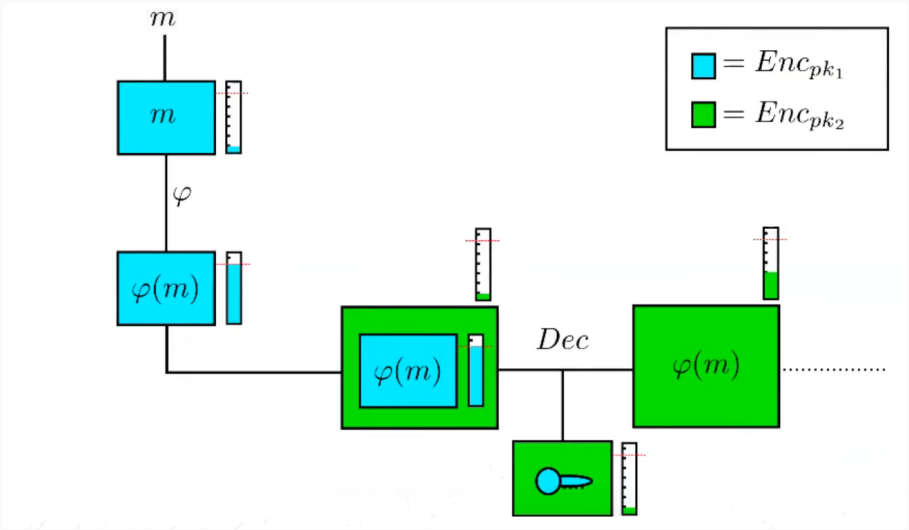
Noise in HE

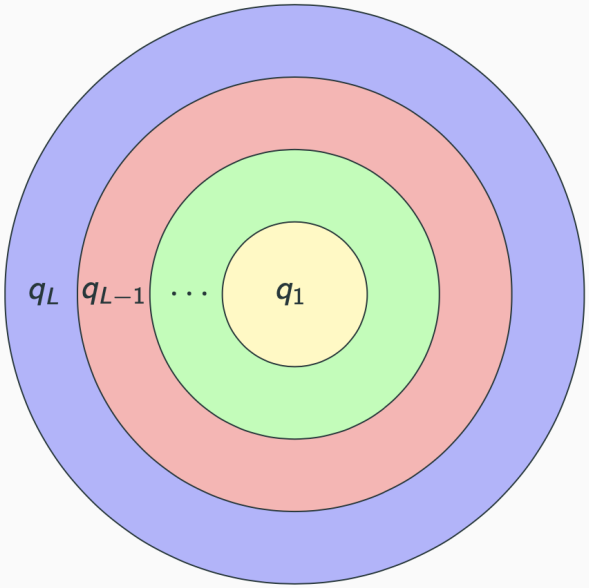
The noise problem, illustrated



Noise management

Two main techniques for managing the noise: bootstrapping and modulus switching (or rescaling).





What is noise and why is it important?

Noise in homomorphic encryption

- All ciphertexts have inherent noise
- Noise grows during homomorphic operations
- If noise too large, decryption will fail

Good understanding of noise growth is essential

- Either need to determine when to bootstrap
- Or need to determine when to do modulus switching
- This enables us to choose appropriate parameters, ideally small ones
- Requiring large parameters remains a major challenge in practical HE

An example of the importance of good noise management

The graph-specific optimisation *lazy rescaling* reduces runtime by 8x.

- Lazy rescaling
  - Only after a fully connected or convolution layer
  - Skip if no multiplications left before decryption

Approximate Homomorphic Encryption over the Conjugate-invariant Ring Duhyeong Kim and Yongsoo Song

Lazy rescaling

Table 5: Impact of lazy rescaling on CryptoNets runtime using $N = 2^{13}, L = 6$, with accuracy 98.95%.

Thread Count	Lazy Rescaling	Runtime	
		Amortized (ms)	Total (s)
1	✗	59.21	242.51 ± 3.69
1	✓	7.23	29.62 ± 0.63
24	✓	0.50	2.05 ± 0.11

nGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data, Fabian Boemer, Rosario Cammarota, Anamaria Costache and Casimir Wierzynski

Worst-case analysis

- Recall the canonical embedding $\sigma(a) = (a(\zeta_i))_{i \in (\mathbb{Z}/N\mathbb{Z})^*}$
- Use the Canonical embedding norm: $\|a\|^{\text{can}} = \|\sigma(a)\|_\infty$
- Variance of $\sigma(a)$ is nV_a

We use

$$\|\mathbf{a}\|^{\text{can}} \leq \mathbf{6}\sqrt{\mathbf{n}}\sqrt{\mathbf{V_a}}\,, \tag{1}$$

and the following facts:

$$V_{a+b} = V_a + V_b$$
$$V_{\gamma a} = \gamma^2 V_a$$
$$V_{ab} = nV_aV_b.$$

C. Gentry, S. Halevi, N. P. Smart. Homomorphic evaluation of the AES circuit. *CRYPTO*, 2012.

BGV results: heuristic bounds vs. observed HElib noise growth

Enc			Add			Mult			ModSw		
P	I	$\overline{x}$	P	I	$\overline{x}$	P	I	$\overline{x}$	P	I	$\overline{x}$
34.0	35.0	41.1	33.0	34.0	40.2	14.0	17.0	26.0	-	-	-
88.0	89.0	97.9	87.0	88.0	97.0	67.0	70.0	82.4	38.0	39.0	38.1
196	197	209	195	196	209	174	177	194	146	147	150
415	416	433	414	415	432	392	395	416	365	366	373

Table 1: The observed mean $\overline{x}$ of the noise budget in HElib ciphertexts in 10000 trials, with estimates of the noise budget obtained using Iliashenko heuristic analysis I and previous heuristic analysis P. Each row corresponds to a parameter set with $n \in \{2048, 4096, 8192, 16384\}$.

Average-case analysis for CKKS

- In CKKS, there are two sources of precision loss
  - Encoding noise
  - Encryption noise that never gets removed
- There is a way to detangle the two and provide an average-case analysis
- Track the noise variance all the way through a circuit via the Central Limit Theorem (CLT)
- Bound it at the end

Noise in the ring

$\log(N)$	$\log(q)$	Average	CLT
Addition noise.			
13	109	10.88	11.40
14	219	11.44	11.93
15	443	12.00	12.45
Multiplication noise.			
13	109	17.31	18.69
14	219	18.38	19.72
15	443	19.43	20.75

Table 2: Average bits of noise observed in the ring over 1000 trials in HEAAN, for $\alpha = 0.0001$ and $\Delta = 2^{40}$.

On the precision loss in approximate homomorphic encryption; Anamaria Costache, Benjamin R. Curtis, Erin Hales, Sean Murphy, Tabitha Ogilvie, and Rachel Player

Results in the complex space

$\log(N)$	$\log(q)$	Average	CLT
Addition, complex error.			
13	109	-21.92	-22.55
14	219	-20.72	-21.52
15	443	-19.70	-20.49
Multiplication, complex error.			
13	109	-23.17	-21.51
14	219	-21.68	-19.92
15	443	-20.13	-18.72

Table 3: Average bits of error observed in the message space over 1000 trials in HEAAN, for $\alpha = 0.0001$ and $\Delta = 2^{40}$.

On the precision loss in approximate homomorphic encryption; Anamaria Costache, Benjamin R. Curtis, Erin Hales, Sean Murphy, Tabitha Ogilvie, and Rachel Player

Evaluating functions homomorphically

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition

---

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning

---

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning
  - The activation function used in Neural Networks is typically non-linear

---

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning
  - The activation function used in Neural Networks is typically non-linear
  - Examples include ReLu, $f(x) = \max(0, x)$

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning
  - The activation function used in Neural Networks is typically non-linear
  - Examples include ReLu, $f(x) = \max(0, x)$
- Typically, we would approximate the function to evaluate

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning
  - The activation function used in Neural Networks is typically non-linear
  - Examples include ReLu, $f(x) = \max(0, x)$
- Typically, we would approximate the function to evaluate
  - Usually via a Taylor Series approximation

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning
  - The activation function used in Neural Networks is typically non-linear
  - Examples include ReLu, $f(x) = \max(0, x)$
- Typically, we would approximate the function to evaluate
  - Usually via a Taylor Series approximation
  - CryptoNets uses $f(x) = x^2$, achieves 99% accuracy

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

What functions can we evaluate homomorphically?

- CKKS/ BGV/ BFV schemes are naturally limited to multiplication and addition
- This can present challenges in some applications, such as Machine Learning
  - The activation function used in Neural Networks is typically non-linear
  - Examples include ReLu, $f(x) = \max(0, x)$
- Typically, we would approximate the function to evaluate
  - Usually via a Taylor Series approximation
  - CryptoNets uses $f(x) = x^2$, achieves 99% accuracy
  - CHET uses $f(x) = ax + b$, for $a, b \in \mathbb{R}$, achieves 81.5% accuracy (down from 84%)

CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy; Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing

Chet: an optimizing compiler for fully-homomorphic neural- network inferencing; Dathathri, R., Saarikivi, O., Chen, H., Laine, K., Lauter, K., Maleki, S., Musuvathi, M., Mytkowicz, T

Machine Learning Applications

Because of the limitations specified above, there are roughly two approaches:

- Modify the network to make it HE-friendly, and re-train

Side note: we mainly consider Neural Networks (NN) in this conversation, but other scenarios are more achievable for FHE [1].

---

[1]: Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption; Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza

Machine Learning Applications

Because of the limitations specified above, there are roughly two approaches:

- Modify the network to make it HE-friendly, and re-train
  - Could lead to much better accuracy and overall performance as the network is now optimised for HE

Side note: we mainly consider Neural Networks (NN) in this conversation, but other scenarios are more achievable for FHE [1].

---

[1]: Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption; Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza

Machine Learning Applications

Because of the limitations specified above, there are roughly two approaches:

- Modify the network to make it HE-friendly, and re-train
  - Could lead to much better accuracy and overall performance as the network is now optimised for HE
- Modify the scheme or protocol

Side note: we mainly consider Neural Networks (NN) in this conversation, but other scenarios are more achievable for FHE [1].

---

[1]: Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption; Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza

Machine Learning Applications

Because of the limitations specified above, there are roughly two approaches:

- Modify the network to make it HE-friendly, and re-train
  - Could lead to much better accuracy and overall performance as the network is now optimised for HE
- Modify the scheme or protocol
  - No need to re-train your network, which is potentially a huge overhead

Side note: we mainly consider Neural Networks (NN) in this conversation, but other scenarios are more achievable for FHE [1].

---

[1]: Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption; Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza

Machine Learning Applications

Because of the limitations specified above, there are roughly two approaches:

- Modify the network to make it HE-friendly, and re-train
  - Could lead to much better accuracy and overall performance as the network is now optimised for HE
- Modify the scheme or protocol
  - No need to re-train your network, which is potentially a huge overhead
- Some scenarios are easier than others (encrypted data vs encrypted model)

Side note: we mainly consider Neural Networks (NN) in this conversation, but other scenarios are more achievable for FHE [1].

---

[1]: Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption; Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza

Machine Learning Applications

Because of the limitations specified above, there are roughly two approaches:

- Modify the network to make it HE-friendly, and re-train
  - Could lead to much better accuracy and overall performance as the network is now optimised for HE
- Modify the scheme or protocol
  - No need to re-train your network, which is potentially a huge overhead
- Some scenarios are easier than others (encrypted data vs encrypted model)
- Training in general is an open problem (backpropagation)

Side note: we mainly consider Neural Networks (NN) in this conversation, but other scenarios are more achievable for FHE [1].

---

[1]: Privacy-preserving semi-parallel logistic regression training with Fully Homomorphic Encryption; Sergiu Carpov, Nicolas Gama, Mariya Georgieva, and Juan Ramon Troncoso-Pastoriza

Interlude: Iterative Algorithms and Approximate Encryption

Iterative algorithms and approximate encryption

- In an exact scheme, we would only need to worry about correctness (and perhaps parameter selection)
- In the clear, we would exactly calculate the $(n+1)^{\text{th}}$ estimate from the n^{th} to obtain a more accurate approximation
- With approximate encryption, the additional noise incurred by the $(n+1)^{\text{th}}$ calculation may mean that calculating the $(n+1)^{\text{th}}$ estimate doesn't provide an advantage over the n^{th} estimate
    - In fact, it may mean that your precision starts worsening instead of improving

Iterative algorithms and approximate encryption

For a given set of parameters, function, and data set, we want to identify the point at which the evaluation noise begins to interfere with the accurate bits of the calculated solution.

- Let $A^{(r)} = ||f(\mathbf{z}) - \mathbf{z}^{(r)}||$ be the absolute error associated with iteration r in the clear,
- $\beta^{(r)}$ be a bound on the real noise in $\mathbb{C}^{N/2}$ at iteration r
- Want to identify the point at which

$$\log \beta^{(r)} > \log A^{(r)}$$

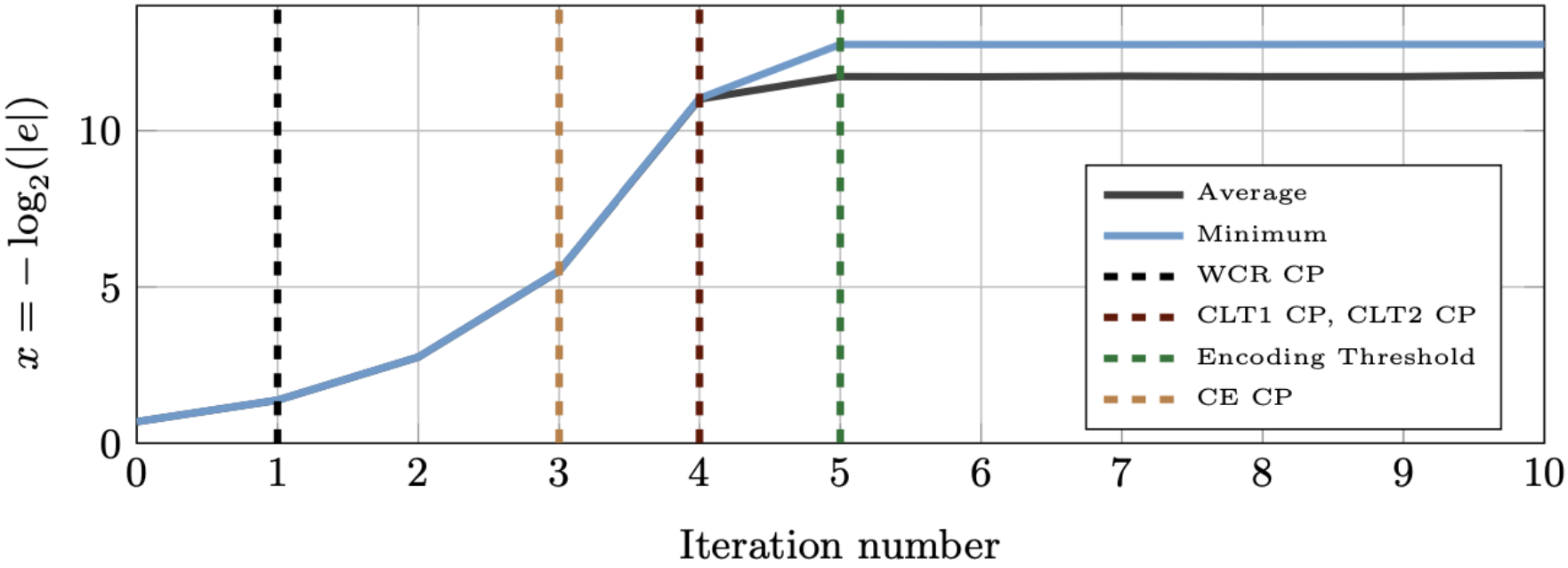
- If this critical point is correctly identified, the maximum possible accuracy that can be guaranteed is

$$-\log A^{(r-1)},$$

as by iteration r encryption noise has begun to interfere with the accuracy $-\log A^{(r)}$

Newton-Raphson

(b) $logN = 14$, $log\Delta = 50$, $a = 1$, $b = 10$.



(c) $logN = 15$, $log\Delta = 35$, $a = 1$, $b = 15$.

Real-life implementation

Deploying FHE

- We have now reached the point where we will “soon” start deploying FHE
- This means we need to start thinking about
  - Lack of CCA security

Deploying FHE

- We have now reached the point where we will "soon" start deploying FHE
- This means we need to start thinking about
  - Lack of CCA security
    - By definition, ciphertexts are malleable

Deploying FHE

- We have now reached the point where we will "soon" start deploying FHE
- This means we need to start thinking about
  - Lack of CCA security
    - By definition, ciphertexts are malleable
    - Trusted Execution Environments (TEE)?

Deploying FHE

- We have now reached the point where we will “soon” start deploying FHE
- This means we need to start thinking about
  - Lack of CCA security
    - By definition, ciphertexts are malleable
    - Trusted Execution Environments (TEE)?
  - Key management?

Deploying FHE

- We have now reached the point where we will "soon" start deploying FHE
- This means we need to start thinking about
  - Lack of CCA security
    - By definition, ciphertexts are malleable
    - Trusted Execution Environments (TEE)?
  - Key management?
    - With eg AES you might use Hardware Security Modules (HSM); not practical for FHE

Deploying FHE

- We have now reached the point where we will "soon" start deploying FHE
- This means we need to start thinking about
  - Lack of CCA security
    - By definition, ciphertexts are malleable
    - Trusted Execution Environments (TEE)?
  - Key management?
    - With eg AES you might use Hardware Security Modules (HSM); not practical for FHE
  - ... ?

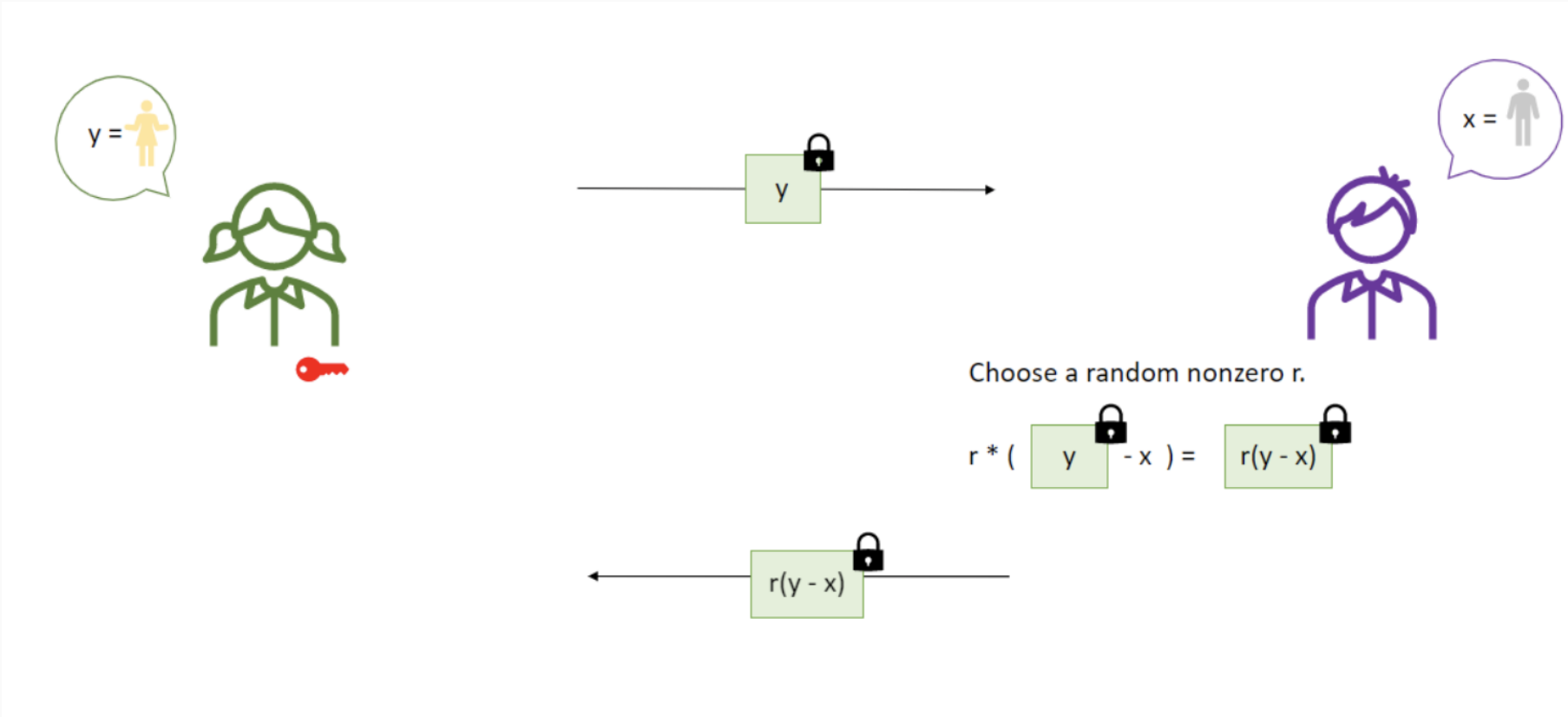
HE as a Primitive

HE as a Primitive

One can also think of HE as a primitive, which enables many constructions.

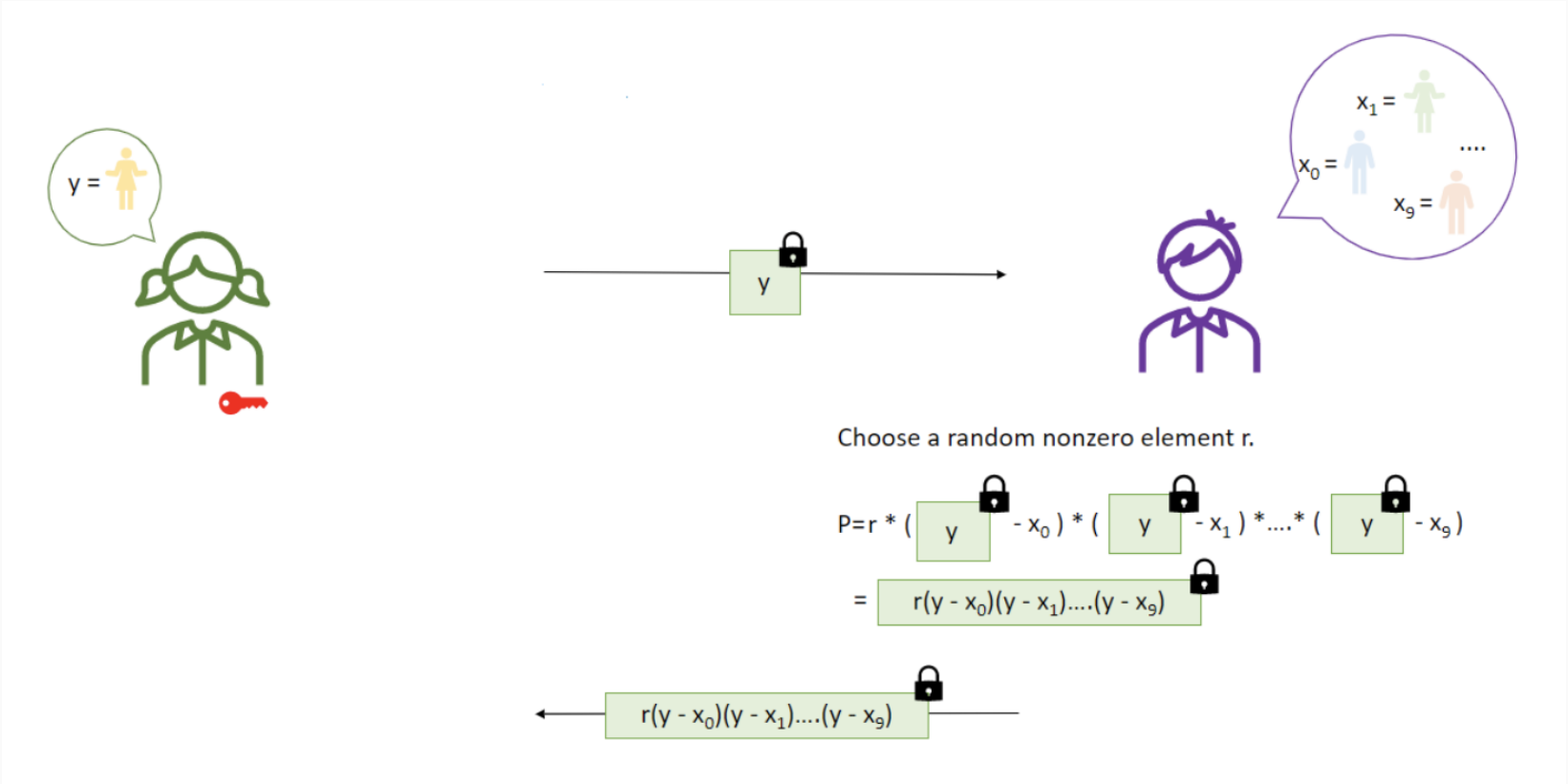
- Multi-Party Computation (MPC)
- Private Set Intersection (PSI)
- Privacy-Preserving Smart Contracts
- Private Set Union
- ...

PSI from FHE



https://bit-ml.github.io/blog/post/private-set-intersection-an-implementation-in-python/

PSI from FHE



---

https://bit-ml.github.io/blog/post/private-set-intersection-an-implementation-in-python/

Conclusion

2010

Dimension n	bit-size t	sparse-subset-size s	big-set size S	big-set ratio R
512	380	15	512	2^{26}
2048	380	15	512	2^{102}
8192	380	15	547	2^{381}
32768	380	15	2185	2^{381}

Dimension n	bit-size t	# of ctxts in PK $(s \cdot c)$	PK size $\approx s \cdot c \cdot \lvert d \rvert$	keyGen	Recrypt
512	380	690	17 MByte	2.5 sec	6 sec
2048	380	690	69 MByte	41 sec	32 sec
8192	380	705	284 MByte	8.4 min	2.8 min
32768	380	1410	2.25 GByte	2.2 hour	31 min

Table 3: Parameters of the fully homomorphic scheme, as used for the public challenges.

Implementing Gentry's Fully Homomorphic Encryption Scheme, Craig Gentry and Shai Halevi `https://eprint.iacr.org/2010/520.pdf`.

2019

Method	Acc. (%)	Lat. (s)	Thput. (im/s)	Protocol	Comm. (MB/im)
LoLa [10]	98.95	2.2	0.5	HE	
FHE-DiNN100 [8]	96.35	1.65	0.6	HE	
CryptoNets [24]	98.95	250	16.4	HE	
Faster CryptoNets [16]	98.7	39.1	210	HE	
nGraph-HE [6]	98.95	16.7	245	HE	
CryptoNets 3.2 [10]	98.95	25.6	320	HE	
nGraph-HE2	98.95	2.05	1,998	HE	
Chameleon [36]	99	2.24	1.0	HE-MPC	5.1
MiniONN [31]	98.95	1.28	2.4	HE-MPC	44
Gazelle [27]	98.95	0.03	33.3	HE-MPC	0.5
nGraph-HE2-ReLU	98.62	0.69	2,959	HE-MPC	0.03

nGraph-HE2: A High-Throughput Framework for Neural Network Inference on Encrypted Data, Fabian Boemer, Rosario Cammarota, **Anamaria Costache** and Casimir Wierzynski `https://eprint.iacr.org/2019/947.pdf`

Conclusion

- Some spectacular advances have been made since the first scheme in 2009
- In particular, this means that we are realistically looking at deplying FHE in the industry in the next "few years"
- There are some very exciting open problems, both theoretical and practical
- ... join us :)

Thank you!